SEE FOLLOWING NOTE

VOIR NOTE CI-APRÈS

PARTS OF THE ORIGINAL DOCUMENT ARE OF POOR REPRODUCTIVE QUALITY.

MICROMEDIA LIMITED HAS ATTEMPTED TO PRODUCE THE BEST POSSIBLE COPY.

CERTAINES PARTIES DU DOCUMENT ORIGINAL SONT DE PIÈTRE QUALITÉ DE REPRODUCTION.

MICROMEDIA LTEE A ESSAYEE DE PROCURER LE MEILLEUR EXEMPLAIRE POSSIBLE.

CA1
DTR290-
90
A71
472092

41.D

TP 10770E

ATCS SYSTEM SAFETY VALIDATION PROGRAMS

prepared for
Transportation Development Centre
Policy and Coordination Group
Transport Canada
Under the Visiting Experts Program

November 1990

by

Dr. H. T. Mouftah
Department of Electrical Engineering
Queen's University
Kingston, Ontario

Library and Information Centre /
Centre de Documentation (CSLA)
RECEIVED - REÇU LE
JUL 9 1991
Transport Canada / Transports Canada

Research Report No. 90-6

472092

DDN 467310

92-01004/

The contents of this report reflect the views of the author and not necessarily the official view or opinions of the Transportation Development Centre of Transport Canada.

Un sommaire en français de ce rapport est inclus avant la table des matières.

Transport Canada Transports Canada

PUBLICATION DATA FORM

1. Transport Canada Publication No.
TP 10770E

2. Project No.
7454

3. Recipient's Catalogue No.

4. Title and Subtitle
ATCS System Safety Validation Programs

5. Publication Date
November 1990

6. Performing Organization Document No.
90-6

7. Author(s)
H.T. Mouftah

8. Transport Canada File No.
ZCD 1841-7-2 Mouftah

9. Performing Organization Name and Address
Department of Electrical Engineering
Queen's University
Kingston, Ontario
K7L 3N6

10. DSS File No.

11. DSS or Transport Canada Contract No.
AR-0925

12. Sponsoring Agency Name and Address
Transportation Development Centre (TDC)
200 Rene Levesque Blvd. West
Suite 601, West Tower
Montreal, Quebec
H2Z 1X4

13. Type of Publication and Period Covered

14. Project Officer
S. Vespa

15. Supplementary Notes (Funding programs, titles of related publications, etc.)
Research project conducted under the Visiting Experts Program

16. Abstract

Because ATCS is an automated system based essentially on the extensive use of computers and communications, special safety considerations must be applied at the different levels of design, development and life cycle of ATCS-type systems. Otherwise all the benefits that ATCS may provide, including improved safety, can be jeopardized. Some railways are introducing new train control systems that may not necessarily follow the existing ATCS standards. Also, ATCS specifications will be continually evaluated and updated as better technology comes along. In view of this, and since ATCS equipment may be supplied by several different manufacturers, a mechanism is needed to ensure that all equipment vital to safety not only meets the specifications but also is compatible with equipment from other manufacturers. Furthermore, to ensure system safety, the specifications themselves must be verified and validated.

Some railways and suppliers have made significant efforts to develop ATCS system validation programs. These efforts are identified in this report, but, in general, they fall short of meeting all the main objectives of an ATCS system validation program.

This report proposes a unified approach for an ATCS system safety validation program, based on the ATCS System Integration Simulator/Emulator/Tester (SISET). It is recommended that the Government promote the establishment of a SISET Laboratory in Canada by the railways, their associations and supplier industry.

17. Key Words
Advanced train control, safety, simulation, rail operations, validation, testing.

18. Distribution Statement
Limited number of copies available from the Transportation Development Centre.

19. Security Classification (of this publication)
Unclassified

20. Security Classification (of this page)
Unclassified

21. Declassification (date)
--

22. No. of Pages
35

23. Price
--

TDC/CDT 79-005
Rev 88

Canada

Transports Canada Transport Canada

FORMULE DE DONNÉES POUR PUBLICATION

1. Nº de la publication de Transports Canada

TP 10770E

2. Nº de l'étude

7454

3. Nº de catalogue du destinataire

4. Titre et sous-titre

ATCS System Safety Validation Programs

5. Date de la publication

Novembre 1990

6. Nº du document de l'organisme exécutant

90-6

7. Auteur(s)

H.T. Mouftah

8. Nº de dossier — Transports Canada

ZCD 1841-7-2

9. Nom et adresse de l'organisme exécutant

Department of Electrical Engineering
Queen's University
Kingston (Ontario)
K7L 3N6

10. Nº de dossier — ASC

11. Nº de contrat — ASC ou Transports Canada

AR-0925

12. Nom et adresse de l'organisme parrain

Centre de développement des transports (CDT)
200 ouest, boul. René-Lévesque
Tour ouest, suite 601
Montréal (Québec)
H2Z 1X4

13. Genre de publication et période visée

14. Agent de projet

S. Vespa

15. Remarques additionnelles (Programmes de financement, titres de publications connexes, etc.)

Recherche au titre du Programme des experts invités.

16. Résumé

Etant donné que les SACT (systèmes avancés de commande des trains) sont les automatismes faisant largement appel à l'électronique et à la télécommunication, la sûreté de leur fonctionnement doit être garantie à tous les stades de leur évolution : conception, développement et vie opérationnelle. Faute de quoi, les avantages qu'ils offrent et la sécurité accrue qu'ils promettent risquent d'être définitivement compromis. D'autre part, certaines sociétés ferroviaires mettent en service des automatismes de commande obéissant à des spécifications qui peuvent différer de celles qui ont présidé à l'étude des SACT. D'autre part, on sait que ces derniers, en plus d'être évolutifs et perfectibles, sont de provenance aussi variée qu'il y a de fournisseurs. Il est donc nécessaire de disposer d'un mécanisme qui non seulement permette de vérifier la conformité des systèmes d'exploitation aux spécifications prévues, mais aussi de garantir leur interopérabilité, quelle qu'en soit la provenance. Enfin, dans l'intérêt même de la sécurité, les spécifications elles-mêmes doivent être constamment vérifiées quant à leur cohérence et validité.

Des sociétés ferroviaires et des fournisseurs ont fourni d'importants efforts à l'élaboration d'un programme de validation SACT. Le présent rapport en fait état, mais force est de constater qu'aucun d'eux ne remplit toutes les conditions auxquelles un tel programme doit satisfaire.

Le présent rapport propose de donner une approche cohérente à la réalisation d'un programme de validation de la sûreté de fonctionnement des SACT, réalisé à partir d'un matériel permettant la simulation, l'émulation et le contrôle des fonctions requises (SISET). Il est donc recommandé que le gouvernement fixe, comme un des objectifs de la R&D, la promotion active d'un laboratoire SISET au Canada, qui serait établi par le secteur privé : sociétés, associations et fournisseurs ferroviaires.

17. Mots clés

Systèmes avancés de commande des trains, sécurité, simulation, fonctionnement des trains, validation, essai.

18. Diffusion

Le Centre de développement des transports dispose d'un nombre limité d'exemplaires.

19. Classification de sécurité (de cette publication)	20. Classification de sécurité (de cette page)	21. Déclassification (date)	22. Nombre de pages	23. Prix
Non classifiée	Non classifiée	—	35	—

CDT/TDC 79-005
Rev 88

Canada

SUMMARY

The Advanced Train Control System (ATCS) is being developed as a joint project of the Railway Association of Canada (RAC) and the Association of American Railroads (AAR). The project is to develop a computer-based method of controlling railroad operations that is safer and more efficient than traditional wayside signal-based control strategies. ATCS will improve efficiency through more timely and accurate train and work crew location reporting; through improved meet-pass planning; and through improved safety. ATCS will enhance safety by ensuring train separation; by monitoring wayside device status and health; by verifying, via computer, the safety and integrity of all authorities issued to trains and maintenance crews; and by enforcing the authorities and speed limits issued to trains by automatic brake application, if required.

Because ATCS is an automated system based essentially on the extensive use of computers and communications, special safety considerations must be applied at the different levels of design, development and life cycle of ATCS-type systems. Otherwise all the benefits that ATCS may provide, including improved safety, can be jeopardized. Some railways are introducing new train control systems that may not necessarily follow the existing ATCS standards. Also, ATCS specifications will be continually evaluated and updated as better technology comes along. In view of this, and since ATCS equipment may be supplied by several different manufacturers, a mechanism is needed to ensure that all equipment vital to safety not only meets the specifications but also is compatible with equipment from other manufacturers. Furthermore, to ensure system safety, the specifications themselves must be verified and validated.

The functional sophistication, as well as complexity of interaction between elements of the ATCS system demand thorough system integration prior to being placed in service. An ATCS system validation program would be based essentially on an integration testing system. Several railways have already started to test some hardware. However, field testing is not enough and cannot substitute for an integration testing in the laboratory.

Some railways and suppliers have made significant efforts to develop ATCS system validation programs. These efforts are identified in this report, but, in general, they fall short of meeting all the main objectives of an ATCS system validation program, which can be summarized as follows:

(i) To demonstrate that equipment is designed, implemented and operated in conformance with specifications.

(ii) To demonstrate that equipment of different suppliers and different railways are interoperable over dark as well as signalled territories.

(iii) To demonstrate that equipped and non-equipped systems are interoperable over dark as well as signalled territories.

(iv) To verify and validate specifications.

In order to meet these objectives, this report proposes a unified approach for an ATCS system safety validation program, based on the ATCS System Integration Simulator/Emulator/Tester (SISET). It is expected that the development of SISET will cost $5 million Canadian, over a period of 5 years.

It is recommended that one R&D objective of Government be to promote and support the development and deployment of a SISET Laboratory in Canada. Industry should have the primary role and responsibility for such a facility.

SOMMAIRE

Fruit de la collaboration entre l'Association des chemins de fer du Canada et l'Association of American Railroads (AAR), les systèmes avancés de commande des trains (SACT) sont des automatismes de sécurité ferroviaire pilotés par ordinateur, à la fois plus performants et plus sécuritaires que la protection assurée par les actuels systèmes de signalisation latérale. Ils sont plus performants du fait qu'ils permettent de déterminer la position d'un train ou d'une équipe de maintenance avec plus de précision et de rapidité et qu'ils promettent une planification plus serrée des temps de passage et de croisement des trains. Ils sont plus sécuritaires parce qu'ils peuvent: suivre de près la séparation entre trains, diagnostiquer l'état où se trouvent les dispositifs d'exploitation, vérifier la compatibilité des consignes de marche et de maintenance et, enfin, vérifier l'observation des consignes de vitesses et, au besoin, télécommander l'arrêt immédiat du train en infraction.

Or, étant donné que les SACT sont des automatismes faisant largement appel à l'électronique et à la télécommunication, la sûreté de leur fonctionnement doit être garantie à tous les stades de leur évolution: conception, développement et vie opérationnelle. Faute de quoi, les avantages qu'ils offrent et la sécurité accrue qu'ils promettent risquent d'être définitivement compromis. D'une part, certaines sociétés ferroviaires mettent en service des automatismes de commande obéissant à des spécifications qui peuvent différer de celles qui ont présidé à l'étude des SACT. D'autre part, on sait que ces derniers, en plus d'être évolutifs et perfectibles, sont de provenance aussi variée qu'il y a de fournisseurs. Il est donc nécessaire de disposer d'un mécanisme qui non seulement permette de vérifier la conformité des systèmes d'exploitation aux spécifications prévues, mais aussi de garantir leur interopérabilité, quelle qu'en soit la provenance. Enfin, dans l'intérêt même de la sécurité, les spécifications elles-mêmes doivent être constamment vérifiées quant à leur cohérence et validité.

Les SACT sont si sophistiqués et leurs interactions si complexes que la sûreté de leur fonctionnement exige une démarche (systémique) rigoureuse; d'abord parfaire leur intégration à l'intérieur du système et, ensuite, vérifier leur compatibilité avec l'ensemble du système par un programme de validation fondé sur des tests fonctionnels d'ensemble. Des sociétés ferroviaires ont déjà lancé ce processus, mais les essais sur le terrain ne suffisent pas et rien ne remplace les essais de laboratoire.

Des sociétés ferroviaires et des fournisseurs ont consenti d'importants efforts à l'élaboration d'un programme de validation SACT. Le présent rapport en fait état, mais force est de constater qu'aucun d'eux ne remplit toutes les conditions auxquelles un tel programme doit satisfaire et qui sont récapitulées comme suit :

i) Vérifier que les équipements sont conformes aux spécifications, dans leur conception, leur réalisation et leur fonctionnement.

ii) Vérifier l'interopérabilité des équipements de provenance et de destination différentes, tant dans les zones protégées par une signalisation automatique que dans celles qui ne le sont pas.

iii) Vérifier que tant les trains équipés de systèmes SACT que ceux qui ne le sont pas peuvent fonctionner aussi bien dans les zones protégées par une signalisation automatique que dans les autres zones.

iv) Vérifier que les spécifications elles-mêmes sont cohérentes et valides.

Dans ce but, le présent rapport propose de donner une approche cohérente à la réalisation d'un programme de validation de la sûreté de fonctionnement des SACT, réalisé à partir d'un équipement intégré permettant la simulation, l'émulation et le contrôle des fonctions requises (SISET), au coût estimatif de 5 millions de dollars canadiens sur une période de cinq ans.

Il est donc recommandé que le gouvernement fixe, comme un des objectifs de la R&D, la promotion active d'un laboratoire SISET au Canada, dans lequel le secteur privé sera appelé à jouer un rôle prépondérant.

CONTENTS

Page

1. Introduction 1
    1.1 Research Rationale and Objective 1
    1.2 ATCS Development 1
    1.3 Investigation Conducted 5
    1.4 Terminology 5

2. The Need for an ATCS System Safety Validation Program 7
    2.1 Introduction 7
    2.2 ATCS Safety Considerations 7
        2.2.1 Management Requirements 7
        2.2.2 System Requirements 8
        2.2.3 Technical Requirements 9
        2.2.4 Testing Requirements 12
    2.3 Conformability and Interoperability 12

3. Existing ATCS System Safety Validation Programs 13
    3.1 Introduction 13
    3.2 Railways System Safety Validation Programs 15
        3.2.1 Rockwell International's System Integration Laboratory 15
        3.2.2 Arinc's Efforts 16
    3.3 Suppliers' System Safety Validation Programs 16
    3.4 Government Agencies System Safety Validation Programs 16

4. A Unified Approach for an ATCS System Safety Validation Program 17
    4.1 Introduction 17
    4.2 Requirements for a System Safety Validation Program 17
    4.3 The Unified Approach 19

5. Conclusions 22
    5.1 Summary 22
    5.2 Concluding Remarks and Recommendations 23

Bibliography 26

Appendix A Presentation to Transport Canada and other Government Representatives
Appendix B List of Contacts
Appendix C Draft British Standard on Functional Safety of Programmable Electronic Systems
Appendix D Draft British Standard on Software for Computers in the Application of Industrial Safety-Related Systems

LIST OF FIGURES

		Page
Figure 1	ATCS General Layout	2
Figure 2	ATCS Communications Logical Architecture	4
Figure 3	Railways Testing ATCS Systems	14
Figure 4	ATCS SISET Lab Architecture	20

1. INTRODUCTION

1.1 Research Rationale and Objective

The Advanced Train Control System (ATCS) is being developed as a joint project of the Railway Association of Canada (RAC) and the Association of American Railroads (AAR). The project is to develop a computer-based method of controlling railroad operations that is safe and more efficient than traditional wayside signal-based control strategies. ATCS will improve efficiency through improved timeliness and accuracy of train and work crew location reporting; through improved meet-pass planning; and through improved safety. ATCS will enhance safety by ensuring train separation; by monitoring wayside device status and health; by verifying, via computer, the safety and integrity of all authorities issued to trains and maintenance crews; and by enforcing the authorities and speed limits issued to trains by automatic brake application, if required.

However, because ATCS is based essentially on the extensive use of computers and communications, special safety considerations must be applied at the different levels of design, development, testing and commissioning, as well as in the operation of ATCS-type systems, or the benefits that ATCS may provide can be jeopardized. Several railroads are introducing new train control systems that deviate from the preliminary ATCS standards and it is uncertain how successful the ATCS program will be at achieving true industry-wide commonality. Thus there is a need to develop an ATCS system safety validation program to ensure system safety.

It is the purpose of this project to investigate the safety issues involved in the design, development and implementation of ATCS-type systems in the Canadian Railway Industry. In this study the main objective is to identify and evaluate ATCS system safety validation programs.

1.2 ATCS Development

ATCS consists of the following five major subsystems (see Figure 1):

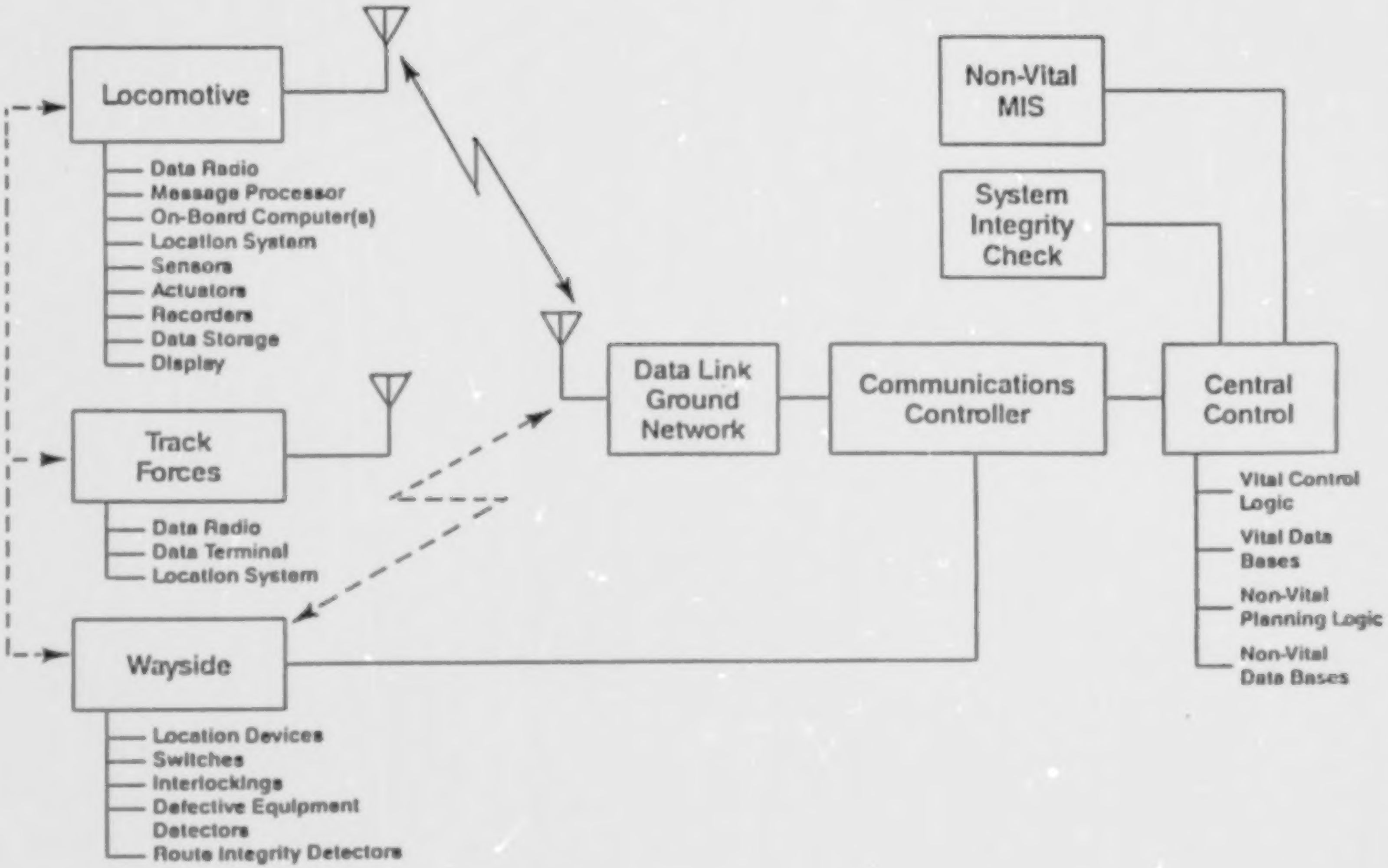
1) Wayside Device Interface Units:

These units provide the capability to control and monitor wayside devices, including switches; interlockings; route integrity devices, such as rock slide and high-water detectors; and train defect detectors, such as high- or wide-load detectors and hot-axle-bearing detectors.

2) Computer-Aided Dispatching:

This subsystem provides automatic train tracking and status monitoring, wayside device monitoring and control, maintenance crew monitoring, and validation and

Figure 1: ATCS GENERAL LAYOUT


Locomotive
Data Radio
Message Processor
On-Board Computer(s)
Location System
Sensors
Actuators
Recorders
Data Storage
Display
Track Forces
Data Radio
Data Terminal
Location System
Wayside
Location Devices
Switches
Interlockings
Defective Equipment Detectors
Route Integrity Detectors
Data Link Ground Network
Communications Controller
Central Control
Vital Control Logic
Vital Data Bases
Non-Vital Planning Logic
Non-Vital Data Bases
Non-Vital MIS
System Integrity Check

protection of each authority to occupy track for both trains and maintenance crews.

3) On-Board Locomotive Computer and Display:

This subsystem provides location determination and reporting using an attached transponder interrogator and odometer. The subsystem also displays authorities, restrictions, slow orders, and other information to the engine driver and enforces safe operation using its brake system interface.

4) Maintenance-of-Way Terminals:

This device enables maintenance crews (track forces) to obtain lineups for forecasts of train arrivals, to apply slow orders, and to request and receive track occupancy permits.

5) Data Communications:

This subsystem provides the mechanism necessary to tie the other four elements together.

The logical architecture of the data communications subsystem is structured according to the reference model for Open Systems Interconnection described in CCITT X.200 [CCI85]. This architecture provides for a seven-layer arrangement of protocols in which each protocol provides services to the layers above and uses the services of the layers below. ATCS uses this logical framework to define protocols; however, it does not use the actual protocols promulgated by CCITT (International Telegraph and Telephone Consultative Committee).

Figure 2 shows the logical components of a communication between a central dispatching computer and a single on-board computer. Normally there would be a large number of such communications in progress at one time. Physically, the data communication subsystem can be partitioned into four major elements:

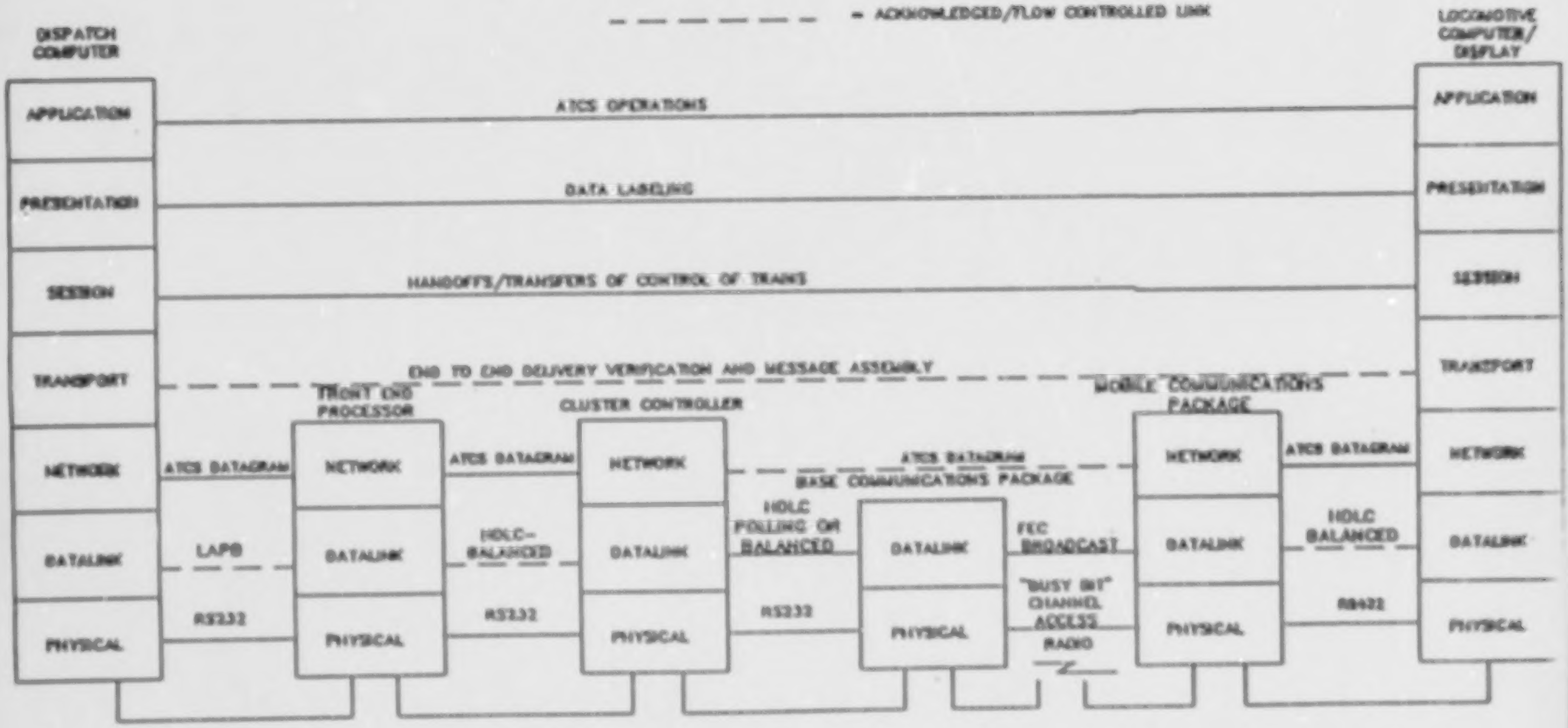
1) The user access protocols that reside in the user devices and allow applications to access and use the communications facilities.

2) The ground network that performs switching and forwarding of message segments (packets).

3) The mobile and wayside networks that provide local connections at wayside devices or on board locomotives.

4) The radio network that provides the connection between the ground network and the various mobile and wayside networks.

Figure 2: ATCS DATA NETWORK BY OSI LAYER


— — — — — — = ACKNOWLEDGED/FLOW CONTROLLED LINK
DISPATCH COMPUTER
APPLICATION
PRESENTATION
SESSION
TRANSPORT
NETWORK
DATALINK
PHYSICAL
ATCS OPERATIONS
DATA LABELING
HANDOFFS/TRANSFERS OF CONTROL OF TRAINS
END TO END DELIVERY VERIFICATION AND MESSAGE ASSEMBLY
FRONT END PROCESSOR
CLUSTER CONTROLLER
BASE COMMUNICATIONS PACKAGE
MOBILE COMMUNICATIONS PACKAGE
ATCS DATAGRAM
LAPB
RS232
HDLC-BALANCED
HDLC POLLING OR BALANCED
FEC BROADCAST
"BUSY BIT" CHANNEL ACCESS
RADIO
HDLC BALANCED
RS422
LOCOMOTIVE COMPUTER/ DISPLAY

The radio network is the least reliable of these elements. This unreliability is caused by the characteristics of the radio channel itself, including intermittent noise and interference from other radio users, multipath interference, and shadowing.

ATCS is a modular system that supports a hierarchy of four differing levels of control. The lowest level, Level 10 - Centralized Route and Block Interlocking System, is a computer aided or monitored block interlocking system without additional on-board or wayside equipment. This is already in common use. Level 20 - Automated Transmission and Display of Instructions, adds on-board and wayside equipment to provide communications. Level 30 - Full Train Tracking, adds continuous monitoring of train location and speed. Level 40 - Full Field interlocking, adds control of the train and its path, including movements through interlockings. Essentially this is the addition of centralized train control to level 30. Automatic driving can also be provided but is not currently anticipated on any North American railways.

1.3 Investigation Conducted

In order to identify and evaluate ATCS system safety validation programs, a comprehensive investigation has been carried out. Several railways, suppliers and government agency representatives in Canada and the United States have been contacted and some have been visited, mainly during the period of January to April 1990. The following railways and industry associations have been visited: Canadian National (Montreal), CP Rail (Montreal), Burlington Northern (Minneapolis), VIA Rail (Montreal), The Railway Association of Canada (Montreal), and the Association of American Railroads (Washington, DC). Suppliers visited were Alcatel SEL Division (Toronto), Dynamic Sciences Limited (Montreal), Vapor (Montreal), and Rockwell International Corporation (Cedar Rapids). The government agencies that have been visited are the Railway Directorate of Transport Canada (Ottawa), The National Research Council (Ottawa), and the Federal Railway Administration (Washington, DC). Appendix B lists names of the people that have been contacted along with their affiliation and telephone numbers.

1.4 Terminology

The following are definitions of some terms that are either used in this report or frequently used in this area [TUR87].

Diverse redundancy:

Use of elements having similar functions but different design and construction.

Fail-operational:

A control system that functions fully even after several component failures.

Fail-passive:

A control system in which safety of operations is not jeopardized by component failures.

Fail-safe:

A control system that either continues to function safely after a failure or lapses into a predefined condition known to be safe.

MTBUF:

Mean time between unsafe failures, a probabilistic specification of the level of safety for a microcomputer-based system.

Safety:

Freedom from unplanned adverse effects in using, operating, and maintaining a system.

Similar redundancy:

Use of elements identical in design and construction as well as in function.

Reliability:

The probability of a system achieving its specified performance over a specified period of time.

Validation:

The process of evaluating a system at the end of its development to ensure compliance with requirements.

Verification:

The process of determining whether the products of a given phase of development fulfill the requirements established in the previous phase.

Vital:

Railroad safety equipment that reverts to a safe state if it can no longer perform its function.

2. THE NEED FOR AN ATCS SYSTEM SAFETY VALIDATION PROGRAM

2.1 Introduction

Because ATCS is an automated system based on the extensive use of computers and communications, special safety considerations must be applied at the different levels of design, development and life cycle of ATCS-type systems, or the benefits that ATCS may provide, including improved safety, can be jeopardized. Some railways are introducing new train control systems that may not necessarily follow the existing ATCS standards. For instance, Burlington Northern has been working with Rockwell International for several years on a version of ATCS it calls Advanced Railroad Electronics System (ARES), which employs the U.S. Department of Defense's Navstar global positioning system (GPS) satellites to locate its trains. Also, Burlington Northern's 2400-baud data link will transmit on frequencies from 160.215 to 161.565 MHz, the range now used by North American railroads for voice communications, rather than the ATCS frequencies around 900 Mhz. ATCS equipment may be supplied by several different manufacturers. ATCS specifications are viewed as "living documents" and will be continually evaluated and updated as better technology comes along [MUR88]. These considerations point to a need for a mechanism to ensure that all equipment vital to safety not only meets the specifications but also is compatible with equipment from other manufacturers. There is also a need to verify and validate the specifications themselves to ensure system safety.

2.2 ATCS Safety Considerations

In this section, safety issues arising from the introduction of ATCS are discussed. Basically, ATCS safety requirements can be identified under four main categories: management requirements, system requirements, technical requirements and testing requirements [MOU88].

2.2.1 Management Requirements

Management requirements include defining safety planning and responsibility, maintenance procedures, operation procedures and personnel management and training.

a) Safety planning and responsibility

The responsibility for the safety aspects of ATCS will ultimately be shared among the manufacturers, the users/operators and the overseeing government agencies. They will be collaborating on the task of ensuring safety of ATCS operations. A safety plan subject to approval by all parties involved in the operation of ATCS should be prepared and executed prior to any operation involving the public. The safety plan must include safety goals and principles, safety analysis methods, operation and maintenance procedures, testing procedures, and associated documentation.

b) Maintenance procedures

A systematic maintenance plan should be prepared with the development of an ATCS. Performance-degrading components should be identified [DOD–a]. Vital components should be tested in operation at regular intervals. New tools, such as a Vehicle Automatic Tester, should be acquired or designed to adapt to the new equipment [BAR87].

c) Operation procedures

In the process of amending the operating rules to include the use of ATCS, special attention has to be given to rules that govern the manner in which the newly introduced equipment will operate, manual override procedures, automatic initiation of braking, as well as changes in procedures for issuing train authority required for ATCS operation. Also, provision must be made for non-ATCS equipped trains to be operated in ATCS territory, as in the initial stages of ATCS deployment.

d) Personnel management

Although ATCS should decrease the complexity of the driver's job and additional special skill training may not be necessary, safe performance of duties should be taught along with other job skills. Safety should be communicated as one of the measures of good performance. Employees should understand company safety policy, employee safety responsibility, and illness and injury-on-the-job reporting procedures. Skill training updates and tests must be performed regularly using updated equipment and facilities.

2.2.2 System Requirements

ATCS system safety requirements include defining ATCS safety goals, safety principles, safety-related modeling and analysis, and operating environment.

a) Safety goals

The overall system safety goal can be defined as follows: the system design should have guaranteed or pre-defined failure modes at both the system and sub-system levels, so that no matter what kind of failure occurs, the consequences will not go beyond the pre-defined failure modes. No other failure modes other than those described above should be allowed. This goal can be further divided into several operations-related subtasks: safe train separation, route integrity assurance, safe interlocking control, train overspeed protection and automatic brake actuation [RAC88b].

b) Safety principles

The overall safety principle is the fail-safe feature. The checked-redundancy principle, which refers to either software or hardware configuration, states that at least one parallel channel must exist in addition to the principal channel for information flow, and the output of the channels should be checked-compared. Any discrepancy will result in corrective actions [SER84], [HER87], [NOR79]. The checked redundancy principle should be used in conjunction with the fail-safe principle and should be subject to verification that the checking process is itself fail-safe or checked redundant and is sufficiently frequent to reduce common mode failure occurrences to a minimum, and that checking failure should be followed by appropriate actions.

It is more appropriate to address the mean-time-to-wrong-side-failure (MTTWF) instead of the mean-time-to-failure (MTTF) that usually characterizes system reliability. MTTWF deals with both the frequency of the failure and the results of the failure [PRA86], [OLS87]. The probability of occurrence of a wrong-side failure which can result in an unsafe condition should be less than 10^{-9} per operating train hour [HER87], [ERI86]. Vital functions/components have to be identified and receive special treatment.

c) Safety-related modeling and analysis

A reliability analysis of the system equipment should be required in the specification issued to manufacturers [BAB86]. It is desirable to have simulated results based on a reliable simulator. A standard reliability model at a functional level should be created. Flow control charts will be able to trace all necessary redundancy paths in the system, including those associated with human factors. Fault tree analysis, as well as failure modes and effects analysis, should also be considered as part of the documented safety and reliability analysis [OLS87], [RAC88b].

d) Operating environment

It is necessary for the equipment to be able to tolerate the extreme variations in temperature and humidity which are typical for the Canadian climate. The mandatory requirement for the exposed equipment should include operating temperature of -40C to +40C and relative humidity of 0% to 100%. The on-board equipment may not necessarily be confined to this standard, due to the availability of on-board air-conditioning. However, in view of a possible air-conditioning system failure, it is recommended that this standard be applied to the on-board equipment as well.

2.2.3 Technical Requirements

ATCS technical requirements include those for the component failure list, as well as on-board, wayside and central equipment, communication systems and software quality assurance.

a) Component failure list

Manufacturers should have carried out a complete analysis and have available a list of possible component failures, their effects, and the consequences of the failures.

b) On-board equipment

Equipment on board the vehicle include the data communication processor(s), the on-board control processor(s) and the locomotive monitoring processor(s). All of these processors perform vital functions which include movement authority management, brake application management, location tracking, and diagnostic data analysis and management. These vital functions require the processors to be fail-safe in design [IEE87], [MIL84], [ROB84].

For this purpose it is recommended that at least two processors be used in parallel, while a standby processor can be optional. Complementary logic design, which means that two processors have the same functional capability but dual logic design, is preferable to avoid common mode failures. If the hardware design is not used, some form of complementary design must exist in the software, meaning two programs with the same functional capabilities but different structures.

A hardware voter with self-checking abilities or some advanced on-line fault diagnosis design should be used to achieve a high degree of safety. In addition, software checks should be used to complement the fail-safe hardware design [AND73]. Redundancy features should also be incorporated in the design of the interface between the hardware and the software.

The keyboard should not be used regularly due to the possibility of keying error. A simple numerical keypad or a special-purpose-type keyboard should be sufficient. The regular keyboard is used only for special instructions. Also, displays that may create possible misinterpretation must be avoided. The redundancy principle should also be applied to this interface as with other vital functions.

c) Wayside equipment

Excluding normal wear and intrinsic failures, the most important consideration for the assurance of continuous operation of wayside equipment is the environmental factor. Climate, vermin, and intentional sabotage must also be accounted for in the design.

d) Central equipment

Environmental factors, size and weight aspect are of less importance here. However, being a vital component, a high level of redundancy is recommended.

e) Communication requirements

Voice communications will continue to perform an important role for train control and supervisory functions, but data communications will expand and become an integral part of the control system. The data link will be the core element of ATCS [ARI85a], [ARI85d], [SHE87a], [SHE87b], [WEH87], [YOS85].

The communications link to the locomotive and track units is a vital link because of the nature of the message [RAC87a]. The overall communications architecture, the protocol design and the data flows should be developed to support a vital link. The communications system must be designed to provide communications integrity across a broad range of potential rates of error caused by noise and other channel limitations [AYE88]. The data formats and addressing schemes will have to be standardized across the industry to ensure compatibility and reduce the probability of interference caused by the reception of data or commands intended for another party. The data link can be made vital through discrete addressing, error detection, retransmission of data blocks that contain errors, message of data block accountability, link integrity checking and control, and full message acknowledgement.

In many instances, a continuous communications link is required because of emergency conditions such as reporting emergency brake applications, being informed of emergency brake applications, remotely initiating a brake application, and providing emergency stop alerts. Many other operating requirements are best satisfied by the ability to communicate with a train at a random time or random location, regardless of movement status. Therefore, radio communication should play an important role for data communications with both locomotives and track forces. For the wayside communications network, radio as well as fibre optics may be more appropriate for deployment, for reasons of safety, reliability, performance and economy [GRO87], [ASH84], [NFK84], [FOR82], [PRR78], [PRR79].

A communication failure list similar to the components failure list [RAC88b] mentioned earlier should be made available.

f) Software quality assurance

A software quality assurance plan should be part of the system safety plan [DYC89], [CSA82], [DOD74], [DOD79], [DOT77], [BEN89], [BSI89a], [BSI89b]. The process of assuring software quality can be achieved by a thorough pre-coding specification, strict coding development practices, and continuous, thorough post-coding testing [DEU82]. A requirement matrix, which correlates all software requirements with the system requirements, the soft/hardware interfaces, and soft/software interfaces should be included in the specification documents [IEE81]. System-level modeling and simulation may also be performed prior to the coding phase [RAM87]. A quantitative software testing assessment method should be followed. Coding development should follow strict practices [GLA79], [GLA86], [MYE75], [TAS78], examples of which are cited in [MOU88]. Post-coding testing should involve testing with an adequate test set which includes all valid as well as invalid input types.

The employment of two parallel development teams is a common practice for higher reliability [RAC88a]. The use of a highly structured language at the design stage is a good practice. The software quality assurance process should be subject to the same safety goals and principles discussed earlier. A software safety plan similar to that of a system safety plan should be prepared with more emphasis on development and life cycle management.

2.2.4 Testing Requirements

Vital functions must be tested thoroughly. A formal test report for each function must be prepared. Sub-systems can be tested through the design of small-scale simulators and mathematical modelling. The in-service use of on-board diagnostic aids can help to test the system thoroughly.

A complete train simulator which is essentially the same as an actual train with video displays that allow realistic train operations may be investigated for train operation testing. Such a simulator can not only increase testing efficiency but can also make training of staff cheaper and faster and allow it to be undertaken earlier. The immediate development of an ATCS testbed facility is very desirable. This will be dealt with in detail in the following sections of this report.

2.3 Conformability and Interoperability

Conformability is defined here as the ability of a system to operate according to or in conformance with a set of established specifications. The ATCS Committee has produced a set of specifications for ATCS design and operation. However, some railways are introducing new train control systems that may not necessarily follow the existing ATCS specifications. In these cases, some safety measures may have to be introduced. For instance, as mentioned earlier, Burlington Northern has been working with Rockwell International on a version of ATCS known as Advanced Railroad Electronics System (ARES), which employs the U.S. Department of Defense's Navstar global positioning system (GPS) satellites to locate its trains [MUR88]. ATCS specifications suggest the use of transponders for this purpose. Also, Burlington Northern's 2400-baud data link will transmit on frequencies from 160.215 to 161.565 Mhz, the range now used by North American railroads for voice communications, rather than the ATCS 4800-baud data link frequencies which are around 900 MHz. Burlington Northern followed this unique route mainly for economic reasons, since the lower frequencies allow it to use its present towers without having to make them taller or install higher-powered transmitters. Another lack of conformability of the ARES system is that the displays in the locomotive cab have been simplified; sensors providing less significant information have been removed.

Thus there is a need to study how the lack or low level of conformability can affect system safety. A system safety validation program should provide the tools to carry out this type of study.

Interoperability is defined as the ability of equipment of different suppliers and different railways to interoperate in all environments [MUR88]. Lack of conformability can lead to a lack of interoperability. For instance, display units of different suppliers may be used by different railways to display incompatible messages or at least non identical types of messages leading to a lack of interoperability. Here, also, safety measures have to be introduced. Thus there is a need of an ATCS system safety validation program that can demonstrate that equipment of different suppliers and different railways are interoperable over dark as well as signalled territories. This is especially important as it will take some time to introduce ATCS by all railways over all territories. It is expected that ATCS systems will coexist along with non-ATCS systems for some time.

3. EXISTING ATCS SYSTEM VALIDATION PROGRAMS

3.1 Introduction

The functional sophistication, as well as complexity of interaction between elements of the ATCS system demand thorough system integration prior to being placed in service. Integration testing should be performed both in a laboratory environment and in the field prior to being placed in actual train control service. An ATCS system validation program would be based essentially on an integration testing system.

Several railways have already started to test some hardware. Figure 3 summarizes some railway field testing of ATCS systems. However, field testing is not enough and cannot substitute for integration testing in the laboratory. Often, problems reported from field testing can be very difficult to re-create in the field. If they involve the movement of trains, we often do not have the luxury of having a train go back and repeat the simulation. Any number of circumstances that led to the problem may be difficult or impossible to repeat. In fact the person observing the problem may not even be able to describe the precise circumstances that caused the problem. In such cases, it is often possible (particularly if a dynamic train simulation is available) to duplicate the situation, theorize a number of possible problem circumstances, and do so on a repetitive basis. This allows the problem and any potential solutions to be evaluated and tested in a fast and economical manner and then be introduced into the field testing environment. Some railways as well as suppliers have also made significant efforts in the development of ATCS system validation programs.

Railway	Involvement
Amtrak (National Railroad Passenger Corp.), Washington, D.C.	Installing on one locomotive Railstar Control Technology's system, which uses Geostar satellites to locate and monitor trains, trucks, and other vehicles
Burlington Northern Railroad Co., Fort Worth, Texas	Testing 230 miles of track and 17 locomotives equipped with a version of ATCS called Advanced Railroad Electronics System (ARES)
Canadian National Railway Co., Montreal	Installing ATCS on 190 miles of track and six locomotives; for more specialized tests, installing test bed on 100 miles of track
CP Rail (Canadian Pacific), Montreal	Has equipped small sections of track with transponders and run some operational tests; was to begin tests of communication system in November 1988
CSX Transportation Inc., Jacksonville, Fla.	Installing ATCS on 50 miles of track and two locomotives; performing some hardware tests
Norfolk & Western Railway Co. and Southern Railway Co., both Norfolk, Va.	In 1986, ran two field tests over 150 miles of track and one over 70 miles of track; have issued a request for proposal for an ATCS to be installed on 250 miles of track and 30 locomotives
Union Pacific Railroad Co., Omaha, Neb.	Testing ATCS on 200 miles of track and two locomotives; also has 260 miles of track and 12 locomotives equipped with communication system to test work-order reporting project

Figure 3. Railways Testing ATCS Systems

3.2 Railways System Validation Programs

Besides field testing testbeds mentioned above, some railways are also developing their own system validation programs, in the form of system integration testing laboratories as well as some software tools. Examples are the Rockwell International's railroad system integration laboratory and Arinc's Communication Interoperability Tester.

3.2.1 Rockwell International's System Integration Laboratory

In 1986, the Railroad Systems Engineering group of the Collins Air Transport Division of Rockwell International Corporation of Cedar Rapids, Iowa, put together a system integration laboratory for the development of the ARES system for Burlington Northern [BUR88]. The lab is now used to evaluate ARES capabilities in train location, data management, locomotive systems analysis, energy management, and rail operations control. Rockwell International claims that this lab possesses the simulation and integration capabilities necessary to ensure the proper operation and safety of ATCS-like systems. To achieve this, it equipped its engineering facility with: complete complements of locomotive equipment, cab mockup with dynamic simulation capabilities, a data communication network, wayside interface units, a full complement of central office computers including a dispatcher and a supervisor workstation, and substantial electronic and computer test instrumentation.

A data network has been constructed on dual redundant packet-switching computers. It is used to enable the testing of network failure modes and to validate network operation. Also included are base station computers and actual radios, which are used for some tests. Three communicating sets of on-board equipment are provided in the lab. One of these "locomotives" has full dynamics simulation capability. As in the case of the on-board systems, tests with the dispatch computer are conducted with more than one wayside location in operation at the same time. The heart of this lab is a central dispatch system capable of all data communications as well as the normal dispatch functions.

The lab is also equipped with a data acquisition terminal to monitor and report the status of locomotive health and maintenance by querying the on-board health computer for information. An interface to the centralized railroad Management Interface System is also provided and tests are carried out over data communication links to the Management Interface System location.

In addition to the actual ARES system elements, the laboratory is equipped with test equipment such as data message analyzers, automatic data logging and other equipment that can deliberately introduce protocol faults. Finally, the lab also contains two types of real-time dynamic simulation capability. The first is a train dynamics simulator while the second is a railroad network simulator.

3.2.2 Arinc's Efforts

The Association of American Railroads (AAR) and the Railway Association of Canada (RAC) have commissioned Arinc Research Corporation of Annapolis, Maryland to develop some tools for ATCS system integration testing. The first tool was the Communications Interoperability Tester (CIT) that was commissioned in February 1990. The main function of the CIT is to verify through testing that communications components from various vendors adhere to specifications and so ensure that communications in a mixed-vendor network are achievable. The CIT can test layers 1 through 3 (see Figure 2) protocols operating over the ATCS radio link. It can exercise the layers 1 through 3 protocols at other interfaces. The plan is that when a substantial amount of ATCS equipment is available, Arinc, supervised by railroad representatives, will run laboratory tests on all equipment vital to safety, to ensure not only that it meets the specifications but also that it is compatible with equipment from other manufacturers [MUR88].

The CIT focuses only on the validation of parts of the Communication system. However, Arinc is also involved in the development of the Control Flow Validator (CFV) software tool, which is expected to be complete by the end of 1990. The CFV will validate the higher level protocols of ATCS communication system architecture (see Figure 2). Furthermore, there is a proposal by Arinc to develop a System Integration Tester (SIT) by the end of 1992 for the ATCS project (AAR and RAC). Very little information is available about Arinc activities in this area.

3.3 Suppliers' System Safety Validation Programs

Some suppliers are developing custom-made testers for their own subsystems. For instance, Alcatel SEL Division of Toronto has developed its own tester for the on-board processor as well as the wayside unit. This was necessary to ensure proper operation and reliability of the subsystem. However, there was no attempt by suppliers to develop a system integration tester or any comprehensive system safety validation program.

3.4 Government Agencies System Safety Validation Programs

There has been no attempt by any government agency to develop an ATCS system validation program that involves system integration testing. However, there are a few small scale studies for testing some aspects of the ATCS specifications to assist in enhancement of system safety. For instance, at the FRA an ATCS test bed locomotive simulator is being developed. The system is based on a full-function TS-3 simulator of IITRI along with a PC running with a program provided by CN. The goal is to study human behaviour related to the use of an ATCS type of display.

At NRC laboratories in Ottawa, a number of train dynamics tests are carried out on a longitudinal train dynamic simulator built by DSL of Montreal. The simulator, which is housed in a VAX 3100 workstation, can be used to allow realistic real-time evaluation of such functions as positioning, stopping distance authority enforcement, and advanced functions such as suitability of pacing speeds, and train handling assistance to the engineer.

4. A UNIFIED APPROACH FOR AN ATCS SYSTEM SAFETY VALIDATION PROGRAM

4.1 Introduction

There is a need to develop a comprehensive ATCS system safety validation program, as was discussed in detail in section 2.

The main objectives of an ATCS system safety validation program can be summarized in the following points:

(i) To demonstrate that equipment is designed, implemented and operated in conformance with specifications.

(ii) To demonstrate that equipment of different suppliers and different railways is interoperable over dark as well as signalled territories.

(iii) To demonstrate that equipped and non-equipped systems are interoperable over dark as well as signalled territories.

(iv) To verify and validate specifications.

In this section a unified approach to implementing ATCS system safety validation program is presented. It is essentially based on what is called an ATCS System Integration Simulator/Emulator/Tester (SISET). Its main features are that it uses off-the-shelf equipment for software development, simulates ATCS subsystems, performs static and dynamic control flow validation, emulates ATCS subsystems, and tests for system interoperability.

4.2 Requirements for a System Safety Validation Program

The main purpose for the development of an ATCS System Integration Simulator/Emulator/Tester is to validate ATCS system safety. This involves showing that equipment from different suppliers and different railways is operating in conformance to specifications and is interoperable over dark as well as signalled territories. A major requirement for an ATCS system safety validation program is the ability to carry out system integration testing, that is the exhaustive testing and evaluation of the total ATCS system in the configuration that will exist in the field. Included in thorough integration testing is the need for a complete performance evaluation of all system functions. This must include fully-automatic functions, as well as those functions which must be exercised by operators of the system. Other tests which must be performed as part of system integration are related to communication protocols and data communications. Every element of all types of messages must be tested for accuracy and completeness. This must be done at the level of individual bits in each message. Message retry protocol, multipacket message transmission and re-assembly, as well as application layer messages, need to be examined in detail. It is important to generate tests which are intended to confuse the various subsystem elements and determine that these do not cause unsafe or unacceptable situations to develop. Similarly, the algorithms contained within the system, such as the positioning algorithms, must undergo testing designed to confuse them. Test cases must be developed which represent not only normal operating modes, but also those that may not normally be encountered, but which cannot be allowed to cause safety problems, or extensive loss of system operation.

ATCS specifications are viewed as living documents. They will be continually evaluated and updated as better technology comes along. A System Integration Simulator/Emulator/Tester (SISET) should be used to test and verify existing specifications under a variety of normal and abnormal operating conditions. Most importantly, tests must be generated for new specifications, to attempt to confuse the various subsystem elements, and determine that these do not cause unsafe situations. This will establish credibility for newly developed specifications.

Throughout the equipment development process, engineers periodically need to test their designs in a near real-world environment. It is not practical to wait until the development process is nearly complete before testing the design with all of its subsystems.

A SISET is essential for the evaluation of all types of human interface issues. This includes those for the dispatcher, the train crew, and personnel using data acquisition terminals. Often, these evaluations must occur in a dynamic environment to be meaningful. Wherever possible, actual workstations and a cab mockup should be provided to enhance the evaluation and make it as realistic as possible.

Often problems reported from field testing can be very difficult to re-create in the field. If they involve the movement of trains, we often do not have the luxury of having a train go back and repeat the situation. It is often possible, particularly if a dynamic train simulation is available, to duplicate the situation, theorize a number of problem circumstances, and to do so on a repetitive basis. This allows the problem and any potential solutions to be evaluated and tested in a fast and economical manner and then be introduced into the field testing environment.

Parts of formal software verification of safety critical functions often must be performed in a real-world environment that incorporates other system elements interfaced to the system under test. A SISET can serve this purpose.

A SISET can be very useful in assisting accident investigations. Often it is necessary to have a train go back and repeat a situation, theorize a number of possible accident scenarios and observe the outcome. Defining the cause of accidents could help to avoid their repetition and hence improve safety.

Along the line of developing new specifications and solving accident problems, a SISET can help in investigating new regulatory issues. Investigating a case for a new regulation may require a train to go back and repeat a situation under different conditions specified in the proposed new regulation.

The facilities contained in a SISET lend themselves to creating training programs, and also provide the means for such activities as running through published training manuals and procedures.

A fully-equipped SISET can be used to develop enhanced new programs such as the braking algorithm, communication procedures, and operating rules.

4.3 The Unified Approach

The proposed unified approach for an ATCS system safety validation program is based essentially on the ATCS SISET. The SISET lab is built on off-the-shelf equipment along with extensive software development to simulate ATCS subsystem elements in order to perform static and dynamic control flow validation and test system interoperability. Also it should be possible to emulate ATCS subsystems, which implies the ability to run laboratory tests on all actual ATCS equipment that are vital to safety, to ensure not only that it meets the specifications but also that it is compatible with equipment from other manufacturers.

The proposed ATCS SISET Lab Architecture is shown in Figure 4. It is composed essentially of the following components: The Communications Network Simulator, Central Dispatch Computer, Wayside Interface Units, Locomotive On-Board Systems, Train Dynamics Simulator, Railroad Network Simulator, Track Equipment Simulator, Track Forces Data Terminal, Management Information System Interface, Data Acquisition Terminal along with the Network Planner, and Test Equipment Resources.

The heart of the SISET lab is a Communications Network Simulator. This simulator is provided on dual redundant packet-switching computers to enable the development and testing of network failure modes. Also included are base station computers, and actual radio transmitters/receivers are used for some tests. The facility enables testing of network throughput, delay, capacity, evaluation of various layers of communication protocol, as well as certain system algorithms such as that used for tracking of moving vehicles. The data network must be available for all other subsystem testing at all times.

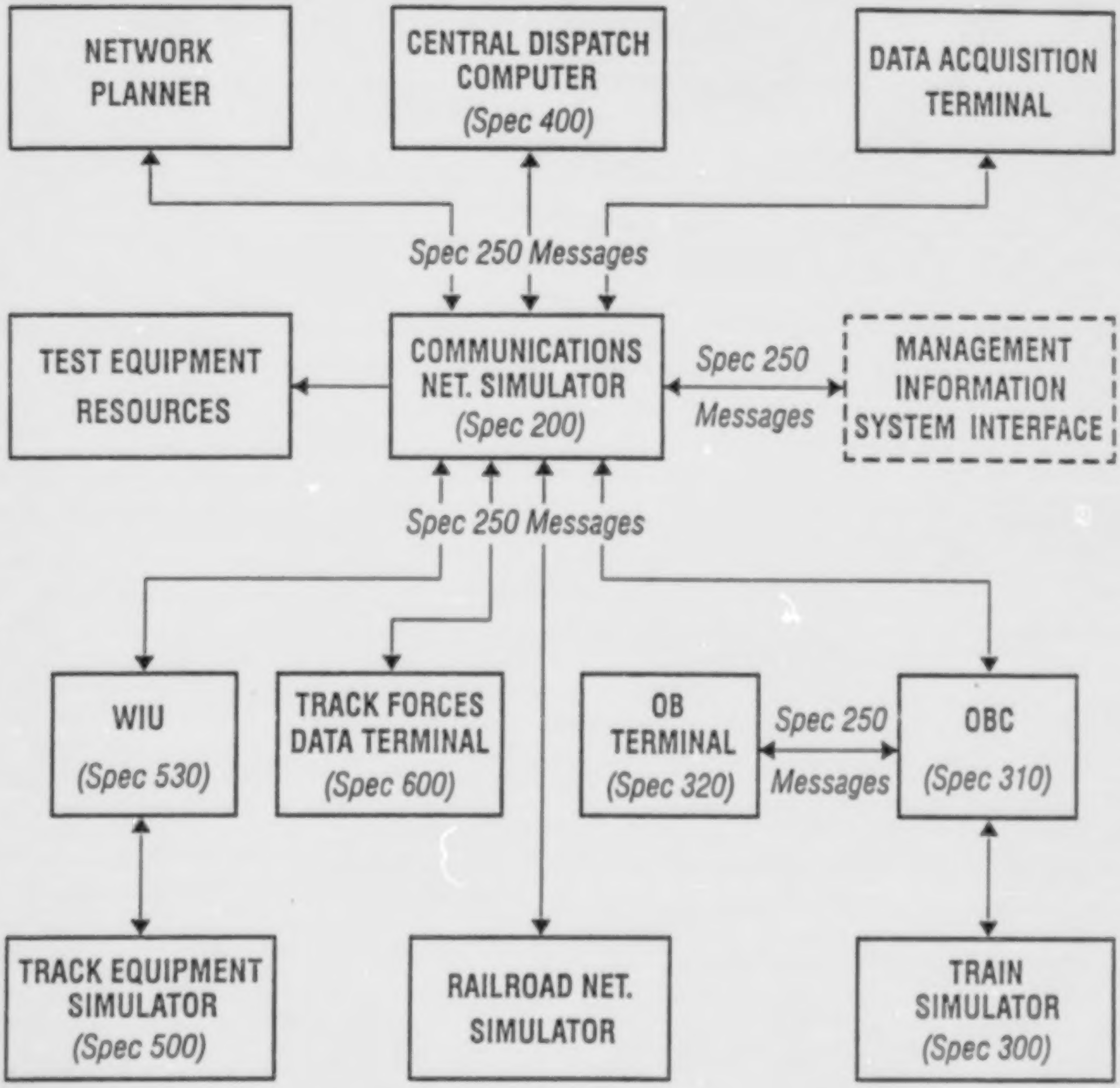

NETWORK PLANNER
CENTRAL DISPATCH COMPUTER
(Spec 400)
DATA ACQUISITION TERMINAL
Spec 250 Messages
TEST EQUIPMENT RESOURCES
COMMUNICATIONS NET. SIMULATOR
(Spec 200)
Spec 250 Messages
MANAGEMENT INFORMATION SYSTEM INTERFACE
Spec 250 Messages
WIU
(Spec 530)
TRACK FORCES DATA TERMINAL
(Spec 600)
OB TERMINAL
(Spec 320)
Spec 250 Messages
OBC
(Spec 310)
TRACK EQUIPMENT SIMULATOR
(Spec 500)
RAILROAD NET. SIMULATOR
TRAIN SIMULATOR
(Spec 300)

Figure 4: **ATCS SISET**

Lab Architecture

The implementation of the Communications Network Simulator must adhere to ATCS specifications 200 while all messages must adhere to ATCS specifications 250.

The heart of any ATCS system is the central dispatch system. All meaningful system integration testing that will include the primary functions of issuing movement authorities, computer aided dispatching, and positioning system must include an on-site central office capability. This machine must be capable of all data communications as well as the normal dispatch functions. A computer along with some communication equipment will be necessary to implement this subsystem which has to adhere to ATCS specifications 400.

Another key element in the total ATCS system is the Wayside Interface Unit (WIU). This subsystem has to follow ATCS specifications 530. Tests with the dispatch computer should be conducted with more than one wayside location in operation at the same time. The WIU inputs and outputs should be connected to hardware test equipment to provide quick visual verification of proper response to commands. A WIU is connected to a Track Equipment Simulator designed to adhere to specifications 500.

More than one communicating set of locomotive on-board equipment is required. Many movement authority functions, as well as hazard warning functions, must be evaluated in an environment that includes more than one vehicle to register, track and communicate with. Three is a reasonable number of sets for evaluating many train control functions in the central dispatch computers. Every set is composed of a checked-redundant computing structure, which adhere to ATCS specifications 310, along with a Mobile Communication Package equipment. One set is equipped with full dynamic simulation capability. Another should be able to emulate the wheel tachometer signal so that, it too, can appear to be travelling at various speeds to the dispatch computer. Two of these equipment sets should have a complete set of test inputs so that locomotive health sensor signals can be emulated, and the resulting messages evaluated as they are sent to the data acquisition terminal. An on-board terminal made to follow specifications 320 is associated with every equipment set.

Several functions contained in ATCS systems cannot be fully evaluated without use of a dynamic environment. Dynamic simulation allows those functions that require moving vehicles to be tested before actually being placed on-site in the field. Two types of dynamic simulation are needed to facilitate this type of integration testing. The first is a train dynamics simulator which has to follow ATCS specifications 300. This machine allows realistic real-time evaluation of such functions as positioning, stopping distance authority enforcement, and advanced functions such as suitability of pacing speeds, and train handling assistance to the engineer. The second type of real-time simulation required is a railroad network simulation. This system simulates an actual railroad environment as viewed by the dispatcher, and the automated processes within the central dispatcher computer. Basically, this simulator illustrates one dispatcher's actual territory. This includes a typical number of trains to control, and should contain CTC, and dark, as well as ATCS types of territory. The (simulated) trains move over the territory, respond to authorities, follow CTC signals and power switches, all as controlled by the dispatcher. This sophisticated capability is crucial to thorough evaluation of train control and display functions in the safety of the lab environment prior to putting the system in the field.

A Data Acquisition Terminal will be used to monitor and report the status of locomotive health and maintenance. Since this terminal becomes a user of the data network, it must be fully tested for proper protocol implementation and interface to the network just as any of the other system elements.

Many ATCS functions will be most efficiently implemented if train information can be automatically entered from railroad Management Information Systems (MIS), and if data from ATCS can be made available on-line to such systems. As with other ATCS system elements, provisions must be made to test these interfaces in the lab before going into service. Since it will not be practical to have MIS computers in the SISET lab, these tests will need to be executed over data communication links to the machine location.

In addition to the actual ATCS system elements, the SISET lab must be equipped with test equipment that allows problem identification. This consists of commercially available data message analyzers that allow viewing of data communication messages as they are transmitted and received. Also, extensive automatic data logging must be available. Equipment must also be provided that can deliberately introduce protocol faults. This can take the form of a standard PC programmed with test cases that are intended to trigger error detection software in the units being tested. To ensure safety, it is important to be able to consistently create these test cases to evaluate any new or modified software under repeatable conditions.

5. CONCLUSIONS

5.1 Summary

Because ATCS is an automated system which is based essentially on the extensive use of computers and communications, special safety considerations must be applied at the different levels of design, development and life cycle of ATCS-type systems. Otherwise all the benefits that ATCS may provide, including improved safety, can be jeopardized. Some railways are introducing new train control systems that may not necessarily follow the existing ATCS standards. Also, ATCS specifications will be continually evaluated and updated as better technology comes along. In view of this fact, and since ATCS equipment may be supplied by several different manufacturers, there is a need for a mechanism to ensure that all equipment vital to safety, not only meet the specifications but also are compatible with equipment from other manufacturers. There is also a need to verify and validate the specifications themselves to ensure system safety.

The functional sophistication, as well as complexity of interaction between elements of the ATCS system demand thorough system integration prior to being placed in service. An ATCS system validation program would be based essentially on an integration testing system. Several railways have already started to test some hardware. However, field testing is not enough and cannot substitute for an integration testing in the laboratory.

Some railways and suppliers have made significant efforts in the development of ATCS system validation programs. These efforts have been identified in this report, but , in general, they fall short of meeting all the main objectives of an ATCS system validation program, which can be summarized in the following points:

(i) To demonstrate that equipment is designed, implemented and operated in conformance with specifications.

(ii) To demonstrate that equipment of different suppliers and different railways is interoperable over dark as well as signalled territories.

(iii) To demonstrate that equipped and non-equipped systems are interoperable over dark as well as signalled territories.

(iv) To verify and validate specifications.

In order to meet these objectives, this report proposes a unified approach for an ATCS system safety validation program. It is essentially based on the ATCS System Integration Simulator/Emulator/Tester (SISET). It is expected that the development of SISET will cost $5 million Canadian, over a period of 5 years.

5.2 Concluding Remarks and Recommendations

1. Main Findings

The following is a summary of the main findings and recommendations from the above investigation, information gathering and evaluation of some existing safety validation programs.

(i) The development of an ATCS system safety validation program is a must.

(ii) Interoperability and conformability are essential for ensuring ATCS system safety.

(iii) Verification and validation of software and communications are necessary for a safe operation throughout ATCS life cycle.

(iv) The development of the ATCS System Integration Simulator/Emulator/Tester (SISET) laboratory is recommended.

(v) Issues still to be resolved include: ownership, location, operation and development.

2. Recommended Government Objective

It is recommended that one R&D objective of Government be to promote and support the development and deployment of a SISET Laboratory in Canada.

Government financial participation is justified in view of the Laboratory's potential to:

(i) Perform safety-related functions throughout the ATCS life cycle; and

(ii) Contribute to enhancing Canada's technological and manufacturing capabilities.

Potential technological and manufacturing contributions are:

a) Maintain Canada's lead in ATCS-related technologies: communications, computer software, real time systems, discrete event simulation, and vehicle dynamics and control.

b) Enhance Canada's ATCS-related manufacturing capabilities, such as those developed at Dynamic Sciences Limited of Montreal, ALCATEL Canada-SEL Division of Toronto, Motorola Information Systems of Toronto, Vapour Canada Inc. of Montreal, Union Switch and Signal of Montreal, etc.

3. Demonstrated Needs of Industry

From the above investigation it is clear that the Canadian railways are interested in the development of a SISET-like laboratory, though they are skeptical about government interference. Also, Canadian railways fear delays that may result from a late development of a SISET laboratory.

On the other hand, suppliers are very interested in the development of a SISET laboratory in Canada. Actually, suppliers are very anxious to cooperate with any party in order to expedite the development of a SISET laboratory. Suppliers are happy with the efforts made to develop comprehensive ATCS system safety validation programs.

4. Potential Participating Government Organizations

Several Canadian Government organizations can contribute to the development of a SISET laboratory in one form or another. The Department of Transportation, Department of Industry, Science and Technology, Department of Communications, as well as the National Research Council are potential participants.

5. Recommended Government Support

(i) Government should promote the establishment of a SISET Laboratory in Canada by the railways, their associations and supplier industry.

(ii) Industry should have the primary responsibility for such a facility.

(iii) Government funding participation in developmental costs could be for SISET's safety-related functions.

(iv) Government participation in annual operating costs could be through provision of facilities to house and operate the laboratory.

BIBLIOGRAPHY

[AKI85] K. Akita, "Practical Use of Computerized Interlocking System "SMILE" in JNR", Japanese Railway Engineering, June 1985, pp. 21-24.

[AKI86] K. Akita, T. Watanabe and H. Nakamura, "Development of Computerized Interlocking System SMILE", Quarterly Reports of R.T.R.I., Vol. 27, No. 3, 1986, pp. 80-83.

[AKI87] K. Akita, H. Nakamura and T. Watanabe, "Solid-State Interlocking in Railway Signalling, SMILE and Micro-SMILE", Quarterly Report of RTRI, Vol. 28, No. 2-4, November 1987, pp. 36-45.

[AND83] H. Anderson, "Experience from the Introduction of ATC in Sweden", Ericsson Review, No. 1, 1983.

[AND73] D.A. Anderson and G. Metze, "Design of Totally Self-Checking Check Circuits for m-Out-of-n Codes", IEEE Transactions on Computers, Vol. C-22, No. 3, March 1973, pp. 263-269.

[ARI85a] ARINC Research Corporation, "Proprietary Identification of Functional Components for Advanced Train Control Systems", ARINC Research Corporation, May 1985.

[ARI85b] ARINC Research Corporation, "Evaluation of Technologies for Advanced Train Control Systems", ARINC Research Corporation, October 1985.

[ARI85c] ARINC Research Corporation, "Selection of System Architecture for ATCS", ARINC Research Corporation, October 1985.

[ARI85d] ARINC Research Corporation, "Basic System Architecture for Advanced Train Control Systems", ARINC Research Corporation, Discussion Paper, December 1985.

[ASH84] M. Ashiya, K. Morihara, H. Ochiai and K. Ohshima, (Mitsubishi Electric Corp.), "Optical Fiber Transmission System for Application in Railway Vehicles", Proceedings of TransPac 85, The Second International Symposium on Advanced Propulsion and Control for Urban Transit, March 4-7, 1984.

[AUC87] J.P. Auclair, "Computer Modularity and Signalling", Rail International, February 1987, pp. 23-37.

[AYE88] R.G. Ayers, "Selection of a Forward Error Correcting Code for the Data

Communications Radio Link of the Advanced Train Control System", Proceedings of the 1988 Joint ASME-IEEE Railway Conference.

[AYE89] R.G. Ayers, "Selection of a Forward Error Correcting Code for the Data Communication Radio Link of the Advanced Train Control System", IEEE Transactions on Vehicular Technology, Vol. 38, No. 4, November 1989, pp. 247-254.

[BAB86] P.S. Babcock IV, "An Introduction to Reliability Modelling of Fault-Tolerant Systems", The Charles Stark Draper Laboratory, Inc., September 1986, 43 pgs.

[BAR83] F.T. Barwell, "Automation and Control in Transportation, 2nd Edition", Pergamon Press, 1983, 390 pgs.

[BAR87] R.E.B. Barnard, "Learning to Live With Micro-Electronic Signalling", GEC Review, Vol. 3, No. 1, 1987, pp. 33-40.

[BEI84] B. Beizen, "Software System Testing and Quality Assurance", Van Nortrand Reinhold, 1984.

[BEL89] T.E. Bell, Issue Editor, Special Report on "Managing Risk in Large Complex Systems", IEEE Spectrum, June 1989, pp. 21-52, (7 papers).

[BEN72] L. Benner Jr., "Safety, Risk and Regulation - Basic Concepts for "Safety" Risk Analysis", Proceedings - Thirteenth Annual Meeting, Transportation Research Forum, Vol. 13, No. 1, November 8-10, 1972.

[BEN89] P.A. Bennett and A.P. Wood (Editors), "Software for Computers in the Application of Industrial Safety-Related Systems, Part One: Proposal for a Standard", International Electromechanical Commission, Document Number: SC65A/WG9/45, June 1989.

[BSI89a] Draft British Standard 0000: Functional Safety of Programmable Electronic Systems: Generic Aspects, British Standards Institution, Document 89/33005 DC.

[BSI89b] Draft British Standard 0000: Software For Computers in the Application of Industrial Safety-Related Systems, British Standards Institution, Document 89/33006 DC.

[BUR88] R.D. Burns, "System Integration and Simulation Requirements for Advanced Train Control Systems", C&S Division, AAR, Committee Reports and Technical Papers, 1988, pp. 341-346.

[CCI85] "Reference Model of Open Systems Interconnection for CCITT Applications", Recommendation X.200, Data Communication Networks Open System Interconnection (OSI), System Description Techniques, Recommendations X.200-X.250, Vol. XIII, 1985, pp. 1-53.

[CHO85] T.S. Chow (Editor), "Tutorial Software Quality Assurance, A Practical Approach", IEEE Computer Society Press, 1985, 497 pgs.

[CIH81] J.R. Cihak, "Guideline for Rail Rapid Transit Equipment, Reliability, Availability and Maintainability Specification", APTA, 1981.

[CNR89] Solicitation for Proposals for CN ATCS Test Bed Locomotive Simulator, Canadian National Railway, Montreal, May 1989.

[COL85] D. Coll, A. Sheikh and K.E. Hancock, "Some Preliminary Thoughts on a Fundamental System and Communications Model and Architecture for the Advanced Train Control System", Lapp-Hancock Associate Limited, Report No. 004/85, August 1985.

[CSA82] Canada Standards Association, "Software Quality Assurance Program, Part 1", Canada Standards Association, Report No. 0317-7874, November 1982, 13 pgs.

[DET87a] P.J. Detmold, "The Purpose of the ATCS Project", IRCA Seminar, Copenhagen, May 20-21, 1987. Rail International, June 1987, pp. 49-53.

[DEU82] M.S. Deutsh, "Software Verification and Validation - Realistic Project Approaches", Prentice-Hall, 1982.

[DIC78] L. Dickinson, "An Evaluation of Railroad Safety", Rail Freight Systems - Research and Conference.

[DOD--a] Department of Defence, "Maintainability Program Requirements", Department of Defence, MIL-STD-470, no date.

[DOD--b] Department of Defence, "Maintainability Verification/Demonstration/ Evaluation", Department of Defence, MIL-STD-471A, no date.

[DOD--c] Department of Defence, "Reliability Design Qualification and Production Acceptance Test: Exponential Distribution", Department of Defence, MIL-STD-781, no date.

[DOD--d] Department of Defence, "Reliability Program for Systems and Equipment Development and Production", Department of Defence, MIL-STD-785, no date.

[DOD74] Department of Defence, "Military Specification - Software Quality

Assurance Program Requirements", Department of Defence, Report No. MIL-S-52779, April 1974, 6 pgs.

[DOD77] Department of Defence, "Military Standard - System Safety Program Requirements", Department of Defence, MIL-STD-882A, 1977.

[DOD79] Department of Defence, "Military Specification - Software Quality Assurance Program Requirements", Department of Defence, Report No. MIL-S-52779A, 1979.

[DOT77] Department of Transportation, "Federal Aviation Administration Standard - Computer Software Quality Program Requirements", Department of Transportation, Report No. FAA-STD-018, May 1977, 5 pgs.

[DYC89] B.M. Dyczkowsky, Editor, "Quality Assurance Program for the Development of Software Used in Critical Applications", Canadian Standards Association, CAN/CSA-Q396.1.1-89 Standard, 1989.

[ERI86] Ericsson, "ATC-System Performance", Document No. ENR/TD 5102, 1986.

[ERI90] R.L. Erickson, A.W. King and W.E. Stace, Guest Editors, IEEE Journal on Selected Areas in Communications Special Issue on Telecommunications Software Quality and Productivity, Vol. 8, No. 2, February 1990, (21 papers).

[FOI86] R.P. Foisy, "Report of the Commission of Inquiry Hinton Train Collision", Supply and Services Canada, December 1986.

[FOR82] G. Forsberg, T. Larsson and K.E. Sundrall, "Optical Transmission Link in ATC Systems", Ericsson Review, No. 2, 1982.

[FOX82] J.M. Fox, "Software and Its Development", Prentice-Hall, Englewood Cliffs, NJ, 1982.

[FRA76] U.S. Department of Transportation, "Special Bibliography - Railroad Safety Research", U.S. Department of Transportation, Federal Railroad Administration, October 1976, 674 pgs.

[FRI84] W. Friesen, (IIT SEL), "Technology Advances in Signalling Systems Utilizing Vital Data Communication", Proceedings of TransPac 85, The Second International Symposium on Advanced Propulsion and Control for Urban Transit, March 4-7, 1984.

[FRI88] W. Friesen, (SEL), "Modern Train Control Technology", Presented at the 67th Meeting of the Transportation Research Board, Washington D.C., 1988.

[GLA79] R.L. Glass, "Software Reliability Guidebook", Prentice-Hall, 1979.

[GLA82] R.L. Glass, "Modern Programing Practices - A Report From Industry", Prentice-Hall, 1982.

[GOL86] A. Goltz, "Safety Demands Formal Software Engineering", Railway Gazette International, July 1986, pp. 495-497.

[GRE90] T.A. Greaves, "ATP Plugs the Gap in Safety Armour", Railway Gazette International, March 1990, pp. 179-183.

[GRI83] A.C. Grinath and P.H. Vess, "Making SQA Word: The Development of a Software Quality System", Quality Progress, July 1983, pp. 18-23.

[GRO87] A.T.M. Grooten, E.J. Bresser and A.G.W.M. Berkers, "Practical Experience With Metal-Free Self-Supporting Aerial Optical Fiber Cable in High Voltage Networks", NKF Kabel B.V., 1987, 11 pgs.

[HAD84] R.J. Hadden and R.L. Kenyon, (Westinghouse), "Operational Experiences With Microprocessor-Based Train Control", Proceedings of TransPac 85, The Second International Symposium on Advanced Propulsion and Control for Urban Transit, March 4-7, 1984.

[HER87] W. Hersey, "Mass Transit Security Regulatory Environment in Canada", Transport Canada, Transportation Development Centre, March 1987, 46 pgs.

[HOL87] C.B. Holden, "The Safety Implications of the Use of Computers in the Railway Industry", Keynote address to COMPRAIL 87 by Major (Retired) C.B. Holden, First Railway Inspectorate, Department of Transport, 1987.

[IEE81] IEEE, "IEEE Standard for Software Quality Assurance Plan", ANSI/IEEE Standard 730-1981.

[IEE87] IEEE, "Fail-Safe Microprocessors", IEEE Spectrum, February 1987.

[JOH83] M.R. Johnson and L. Rohter, "An Advanced Locomotive Simulator: Its Capabilities and Applications", ASME Paper 83-WA-RT-9.

[KOR89] A.L. Kornhauser, "Software Tools for Rail Service Planning, Monitoring and Control", Rail International, October 1989, pp. 11-18.

[LHA88] Lapp-Hancock Associates Limited, "Draft ATCS Subject Bibliography", Lapp-Hancock Associates Limited, June 1988, 37 pgs.

[MAC73] N.A.M. MacKay and B.D. Martin, "Bibliography on Railway Signalling 1960-1972", Canadian Institute of Guided Ground Transport, May 1973, 50 pgs.

[MAC74a] J.A. MacDonald, "A Brief Survey of Rail Safety Technology - Vol. 1", Canadian Institute of Guided Ground Transport, 1974, 63 pgs.

[MAC74b] J.A. MacDonald, "A Brief Survey of Rail Safety Technology - Vol. 2 (Appendix B)", Canadian Institute of Guided Ground Transport, 1974, 250 pgs.

[MAR83] J. Martin and C. McClure, "Software Maintenance - The Problem and Its Solutions", Prentice-Hall, Englewood Cliffs, N.J., 1983.

[MAT80] P.D. Mattison, D.W. Palmer, P.R. Nayak and A.D. Little, "A Report on Investigation Into Rail Passengers Safety", Prepared for U.S. Department of Transportation, Federal Railroad Administration, Office of Research and Development, Washington, D.C.

[MCC89] C. McClure, "The CASE Experience", BYTE, April 1989, pp. 235-244.

[MCG72] C.W. McGuire, "Risk Analysis for Transportation Safety - The Regulator's Viewpoint", Proceedings of Thirteenth Annual Meeting, Transportation Research Forum, Vol. 13, No. 1, November 8-10, 1972.

[MIL84] J.D. Mills, "Processor Based Safety Systems", Paper read in London on December 4, 1984.

[MOO90a] H.G. Moody, "ATCS Components Start Giving Results", Modern Railroads, March 1990, pp. 39-43.

[MOO90b] H.G. Moody, "Panel on Current Status of Advanced Train Control Systems", Presentation at Transportation Research Board Annual Meeting, Washington, D.C., January 1990.

[MOU88] H.T. Mouftah, "Safety Considerations in Implementation of ATCS-Type Systems", Working Paper, CIGGT Report No. 88-4, August 1988.

[MRR86] Modern Railroads, "The Implications of ARES", Modern Railroads, November 1986, pp. 47-48.

[MUR85] H. Murakami, "Application of Microelectronics to Signalling System and Establishment of Technology for Security", Japanese Railway Engineering, December 1985, pp. 16-19.

[MUR88] E.E. Murphy, "All Aboard for Solid State", IEEE Spectrum, December 1988, pp. 42-45.

[MYE75] G.J. Myers, "Reliable Software Through Composite Design", Van Nortrand Reinhold, 1975.

[NET86] Network - Railways of Australia Quarterly, "Sentinels of Safety - Part 1", Network - Railways of Australia Quarterly, Vol. 23, No. 4, 1986, pp. 41-49.

[NET87a] Network - Railways of Australia Quarterly, "Sentinels of Safety - Part 2", Network - Railways of Australia Quarterly, Vol. 24, No. 1, 1987, pp. 43-49.

[NET87b] Network - Railways of Australia Quarterly, "Sentinels of Safety - Part 3", Network - Railways of Australia Quarterly, Vol. 24, No. 2, 1987, pp. 42-49, 62.

[NKF84] NKF Kabel B.V., "Railway Communication Systems", NKF Kabel B.V., 1984, 4 pgs.

[NOR79] D.J. Norton, "Safety by Redundancy", March 1979.

[NTS81] National Transportation Safety Board, "Safety Effectiveness Evaluation of Rail Rapid Transit Safety", National Transportation Safety Board, Report No. NTSB-SEE-81-1, January 1981, 106 pgs.

[OLS87] T. Olson, "Safety Analysis of ATCS", Rockwell International Corp., October 1987, 14 pgs.

[OTA78] Congress of the U.S., Office of Technology Assessment, "An Evaluation of Railroad Safety", Congress of the U.S., Office of Technology Assessment, 1978, 223 pgs.

[PAR87] K.S. Park, "Human Reliability: Analysis, Prediction, and Prevention of Human Errors", Elsevier Science Publishers B.V., 1987, 340 pgs.

[PAR89] T. Parkinson, "Safety Issues Associated With the Implementation of ATCS-Type Systems", Working Paper prepared for the Transportation Development Centre, Transport Canada, August 1989.

[PAW81] R.J. Pawlak, R.M. Snow and M.E. Metcalf, "Annotated Bibliography of Rail Transit Safety, 1975-1980, With Emphasis on Safety Research and Development - Final Report", National Technical Information Service, September 1981, 226 pgs.

[PET86] E.R. Petersen, A.J. Taylor and C.D. Martland, "An Introduction to Computer-Assisted Train Dispatch", Journal of Advanced Transportation, Vol. 20, No. 1, pp. 63-72, 1986.

[PRA86] D.K. Pradhan (Editor), "Fault-Tolerant Computing - Theory and Techniques", Vol. 1 and 2, Prentice-Hall, N.J., 1986.

[PRR78] Progressive Railroading, "Optical-Fiber Transmission in Railroad Future?", Progressive Railroading, January 1978, p. 56.

[PRR79] Progressive Railroading, "Union Pacific First to Use Optical Fibers", Progressive Railroading, January 1979, p. 93.

[PRR89] Staff Editor, "C^3 Comes to the Railroads", Progressive Railroading, September 1989, pp. 44-48.

[RAC84] Railway Association of Canada, "Advance Train Control System - Operating Requirements", Railway Association of Canada, April 1984, 45 pgs.

[RAC87a] Railway Association of Canada, "ATCS Specification 200 - Communications System Architecture", Railway Association of Canada, December 1987, 136 pgs.

[RAC87b] Railway Association of Canada, "ATCS Specification 210 - Mobile Communications Package", Railway Association of Canada, December 1987, 12 pgs.

[RAC87c] Railway Association of Canada, "ATCS Specification 220 - Front End Processor", Railway Association of Canada, December 1987, 10 pgs.

[RAC87d] Railway Association of Canada, "ATCS Specification 225 - Cluster Controller", Railway Association of Canada, December 1987, 10 pgs.

[RAC87e] Railway Association of Canada, ATCS Specification 230 - Base Communications Package", Railway Association of Canada, December 1987, 9 pgs.

[RAC88a] Railway Association of Canada, "ATCS Specification 130 - Software Quality Assurance", Railway Association of Canada, April 1988.

[RAC88b] Railway Association of Canada, "ATCS Specification 140 - Recommended Practices for Safety, Reliability, Maintainability and Quality Assurance", Railway Association of Canada, April 1988.

[RAM87] C.V. Ramamoorthy, A. Prakash, V. Farg, T. Yamaura and A. Bhide, "Issues in the Development of Large, Distributed and Reliable Software", Advances in Computers, Vol. 26, 1987, pp. 393-443.

[REO86] J.A. Reoch, W.J. Moore Ede, G. Pruitt and G.B. Mott, "Advance Train Control Systems", Presentation to the Third International Heavy Haul Railway Conference, October 1986.

[RGI77] Railway Gazette International, "Optic Cable Tried Out Beside Busy Electrified Line", Railway Gazette International, September 1977, p. 359.

[RGI89] Staff Editor, "Safety Spending to Rise by a Third", Railway Gazette

International, April 1989, pp. 235-238.

[ROB84] R. Roberts, "Microprocessor Revolution", Modern Railroads, October 1984, pp. 22-25.

[SCH81] C. Schwier, R.W. Lake and A.W. King, "Evaluation of Railway Safety Projects", CIGGT, Queen's U., Report No. 80-19, March 1981, 54 pgs.

[SER84] O. Serlin, "Fault-Tolerant Systems in Commercial Applications", IEEE Computer, August 1984, pp. 19-30.

[SHE87a] A.U. Sheikh and J. Lee, "ATCS: Advanced Train Control System Communication Architecture and Data Transmission Consideration", 37th IEEE Vehicular Technology Conference, Tampa, Florida, June 1-3, 1987, pp. 214-221.

[SHE87b] A.U. Sheikh and M. Sen, "Design Considerations for ATCS Radio Data Links", IEEE Technical Papers presented at the 1987 IEEE/ASME Joint Railroad Conference, Published by IEEE, 1987, pp. 163-172.

[SHE88] A.U. Sheikh, J. Kraav and W. Marcos, "Mobile Data Link Field Trials: An Evaluation for ATCS", Proceedings of IEEE Vehicular Technology Conference, 1988, pp. 77-82.

[TAS78] D.V. Tassel, "Program Style, Design, Efficiency, Debugging, and Testing, 2nd Edition", Prentice-Hall, 1978.

[TAY78] S.F. Taylor, J.F. Marshall, C.M. Schultz and R.B. Whalen, "Evaluation of Signal/Control System Equipment and Technology - Task 1: Assessment of Signal/Control Technology and Literature Review", U.S. Department of Transportation, Federal Railroad Administration, December 1978, 195 pgs.

[TAY87] G.D. Taylor and D. Klitch, "BC Rail Train Control System (TCS) Test Bed System Development", Prepared by BC Rail for Transportation Development Centre, Transport Canada, Report No. TP 8154 E, January 1987, 51 pgs.

[TOY77] W.N. Toy, "Fault-Tolerant Computing", Advances in Computers, Vol. 26, 1977, pp. 201-279.

[TUR87] D.B. Turner, R.D. Burns and H. Hecht, "Designing Micro-Based Systems for Fail-Safe Travel", IEEE Spectrum, February 1987, pp. 58-63.

[UST76] U.S. Department of Transportation, "Special Bibliograph, Railroad Safety Research", U.S. Department of Transportation, Federal Railroad Administration, Office of Safety Research, Development, and Demonstrations, 1976 XI, 674 pgs.

[UST82] U.S. Department of Transportation, "Safety and Reliability of Automated Guideway Transit Systems - Final Report", U.S. Department of Transportation, Transportation Centre, March 1982, 93 pgs.

[VES87a] S.C. Vespa, "Use of Performance Standards in Railway Safety Regulation", Transportation Development Centre, June 1987, 179 pgs.

[VES87b] S. Vespa, "Report on FRA Public Hearings on Communication Held in Washington on January 28-30, 1987 – Appendix I: Brief Summary of Oral Testimony", TDC Memo, February 23, 1987.

[WEH87] I.L. Wehner, "Progress in the Safety and Reliability of Signalling Technology", Rail International, June 1987, pp. 9-14.

[WEI87] W.W. Weinstein, P.S. Babcock IV and F. Leong, "Safety Analysis of ARES", The Charles Stark Draper Laboratory, Inc., October 1987, 104 pgs.

[WEL88] G. Welty, "ATCS: More Than 'Train Control'", Railway Age, August 1988, pp. 45-49.

[WHI85] W.H. Whitehouse, "New Technology for Signalling and Telecommunications Systems", Rail International, March 1985, pp. 123-131.

[WHI87] L.J. White, "Software Testing and Verification", Advances in Computers, Vol. 26, 1987, pp. 335-391.

[WIL88] G.D. Wilson, "ATCS - A Modular Train Control System", Institution of Railway Signal Engineers, March 1988, 13 pgs.

[YAS84] H. Yasuhara, "Development of Computer-Controlled Interlocking System", Japanese Railway Engineering, Vol. 24, No. 1, 1984, pp. 18-21.

[YOS85] H. Yoshimura, S. Tashiro and I. Okumura, "The Introduction of Micro-Electronics for the Signals", Rail International, February 1985, pp. 131-135.

APPENDIX A

Presentation to Transport Canada
and the Government Representatives

ATCS SYSTEM SAFETY
VALIDATION PROGRAMS

by

H.T. Mouftah
Visiting Expert, TDC
Department of Electrical Engineering
Queen's University at Kingston

OUTLINE

* Introduction
* Objectives
* ATCS Safety Considerations
* Existing ATCS System Validation Programs
* The Unified Approach for an ATCS System Safety Validation Program
* The Need for an ATCS System Integration Simulator/Emulator/Tester
* ATCS SISET Lab Architecture
* Conclusions
* Recommendations

DEFINITIONS

- * Safety:

  Freedom from unplanned adverse effects in using, operating, and maintaining a system.

- * Fail-Safe:

  A control system that either continues to function safely after a failure or lapses into a predefined condition known to be safe.

DEFINITIONS (cont'd)

* Vital:

  Railroad safety equipment that reverts to a safe state if it can no longer perform its function.

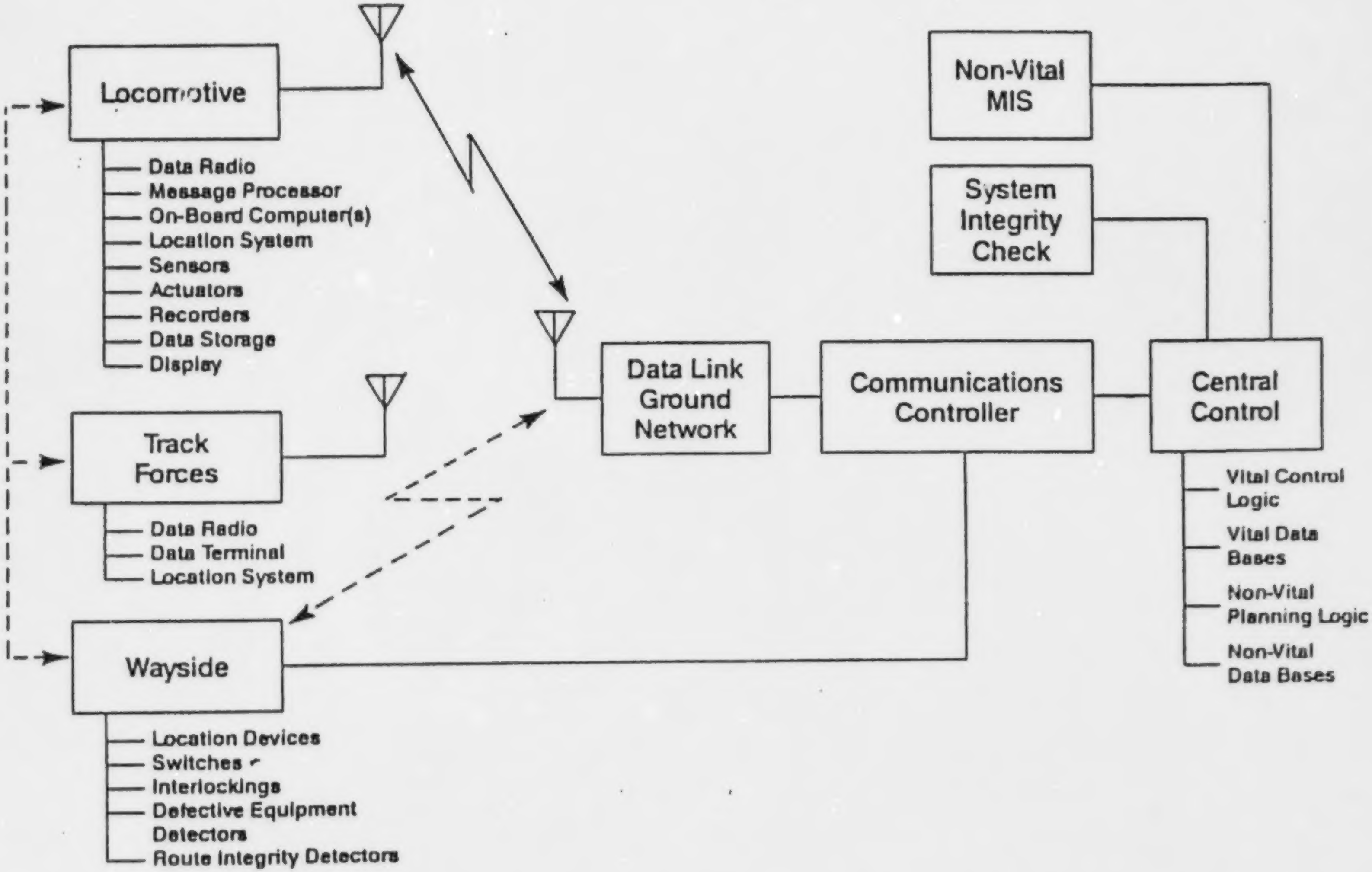
* Reliability:

  The probability of a system achieving its specified performance over a specified period of time.

GENERAL RESEARCH OBJECTIVE

To investigate the safety issues involved in the design, development and implementation of ATCS-type systems in the Canadian Railway Industry.

ATCS GENERAL LAYOUT



Locomotive
- Data Radio
- Message Processor
- On-Board Computer(s)
- Location System
- Sensors
- Actuators
- Recorders
- Data Storage
- Display

Track Forces
- Data Radio
- Data Terminal
- Location System

Wayside
- Location Devices
- Switches
- Interlockings
- Defective Equipment Detectors
- Route Integrity Detectors

Data Link Ground Network

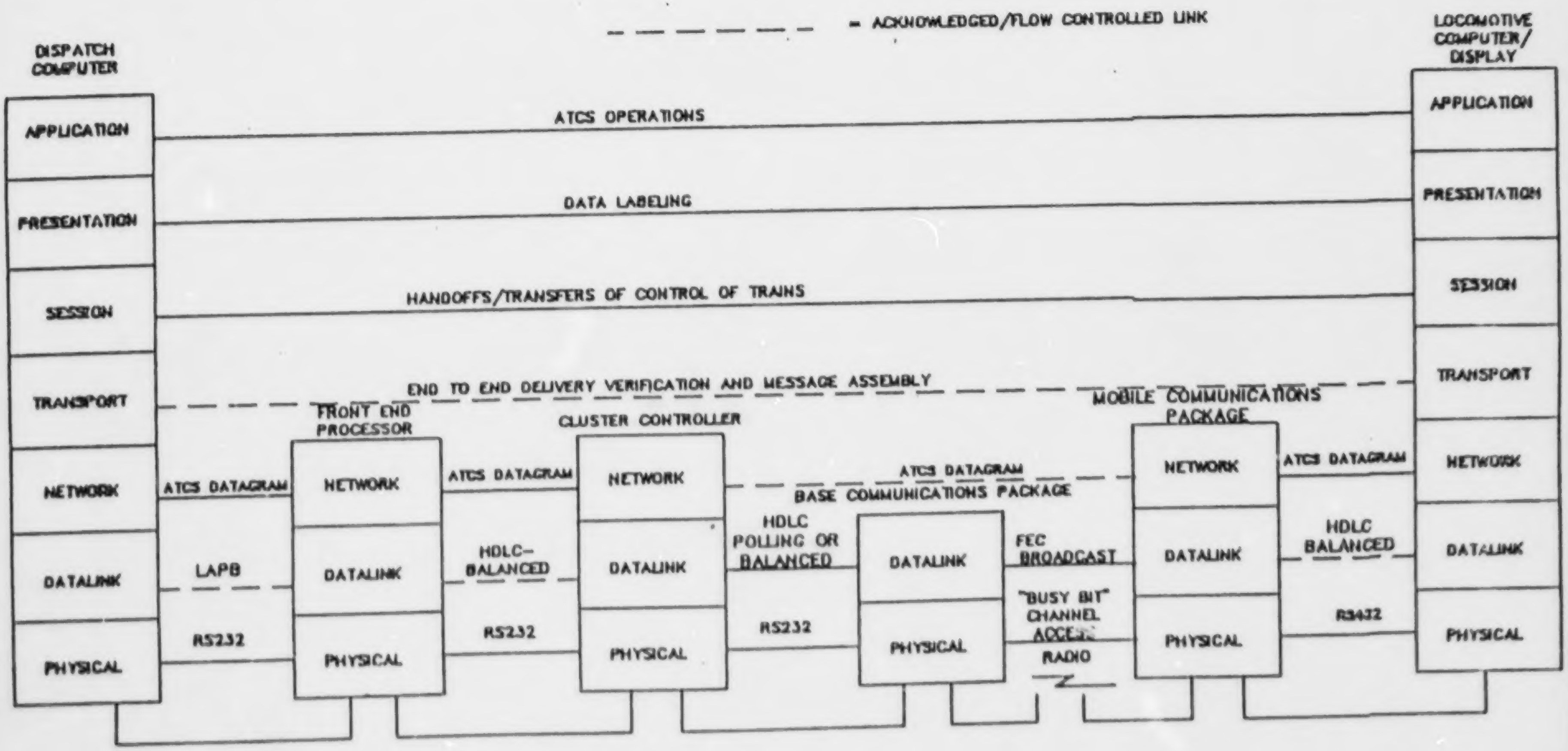
Communications Controller

Central Control
- Vital Control Logic
- Vital Data Bases
- Non-Vital Planning Logic
- Non-Vital Data Bases

Non-Vital MIS

System Integrity Check

ATCS DATA NETWORK BY OSI LAYER


- - - - - - - = ACKNOWLEDGED/FLOW CONTROLLED LINK
DISPATCH COMPUTER
APPLICATION
PRESENTATION
SESSION
TRANSPORT
NETWORK
DATALINK
PHYSICAL
ATCS OPERATIONS
DATA LABELING
HANDOFFS/TRANSFERS OF CONTROL OF TRAINS
END TO END DELIVERY VERIFICATION AND MESSAGE ASSEMBLY
FRONT END PROCESSOR
CLUSTER CONTROLLER
BASE COMMUNICATIONS PACKAGE
MOBILE COMMUNICATIONS PACKAGE
LOCOMOTIVE COMPUTER/ DISPLAY
ATCS DATAGRAM
LAPB
RS232
HDLC-BALANCED
HDLC POLLING OR BALANCED
FEC BROADCAST
"BUSY BIT" CHANNEL ACCESS
RADIO
HDLC BALANCED
RS422

OBJECTIVE

To identify and evaluate ATCS System Safety Validation Programs.

ATCS SAFETY CONSIDERATIONS

* Management Requirements

* System Requirements

* Technical Requirements

* Testing Requirements

ATCS SAFETY CONSIDERATIONS

* Management Requirements
  - Safety Planning and Responsibility
  - Maintenance Procedures
  - Operation Procedures
  - Personnel Management and Training
  - Regulatory Response

ATCS SAFETY CONSIDERATIONS (cont'd)

- * System Requirements
  - Safety Goals
  - Safety Principles
  - Safety-Related Modelling and Analysis
  - Operating Environment

ATCS SAFETY CONSIDERATIONS (cont'd)

* Technical Requirements
  - Component Failure List
  - On-Board Equipment
  - Wayside Equipment
  - Central Equipment
  - Communication Requirement
  - Software Quality Assurance

* Testing Requirements
  - Testing Methods and Procedures
  - Testbed Development

OBJECTIVES OF THE SYSTEM SAFETY VALIDATION PROGRAM

* Demonstrate that equipment is designed, implemented and operated in conformance with specifications.

* Demonstrate that equipment of different suppliers and different railways are interoperable over dark as well as signalled territories.

OBJECTIVES OF THE SYSTEM SAFETY VALIDATION PROGRAM (cont'd)

* Demonstrate that equipped and non-equipped systems are interoperable over dark as well as signalled territories.

* Verify and validate specifications.

GOALS OF THE SYSTEM SAFETY VALIDATION TESTS

* Demonstrate that all vital functions are performed safely under normal operation, and

* Demonstrate that the system responds safely to induced failures in vital equipment.

EXISTING ATCS SYSTEM VALIDATION PROGRAMS

* Rockwell International's System Integration Lab
* Arinc's Efforts:
  - Communication Interoperability Tester (CIT): commissioned February 1990
  - Control Flow Validator (CFV): under development, Dec. 1990 target
  - System Integration Tester (SIT): a proposal, Dec. 1992 target.

EXISTING ATCS SYSTEM VALIDATION PROGRAMS (cont'd)

* Suppliers' Subsystems Custom Made Testers
  - Alcatel/SEL
* Railway Initiatives:
  - Subsystem Simulation and Testing
  - Field Testbed.

ROCKWELL INTERNATIONAL'S RAILROAD SYSTEMS INTEGRATION LABORATORY

- * Formed in 1986 to bring together in one facility, Rockwell's development of ARES
- * Now used to evaluate ARES capabilities in Train Location, Data Management, Locomotive Systems Analysis, Energy Management, and Rail Operations Control

ROCKWELL INTERNATIONAL'S RAILROAD SYSTEMS INTEGRATION LABORATORY

* Composed of the following modules:

- Network Control Computers
- Locomotive Systems Integration Rack
- Train Dynamics and Energy Management Simulator
- Maintenance Terminal
- Rail Operations Control System

COMMUNICATIONS INTEROPERABILITY TESTER (CIT) PURPOSE

Verify through testing that communications components from various vendors adhere to specifications to ensure that communications in a mixed-vendor network are achievable.

The CIT will test layer 1 through 3 protocols operating over the ATCS radio link. The CIT will exercise the layer 1 through 3 protocols at other interfaces.

THE UNIFIED APPROACH FOR
AN ATCS SYSTEM
SAFETY VALIDATION PROGRAM

The ATCS System Integration
Simulator/Emulator/Tester
(SISET)

SISET MAIN FEATURES

* Software Development on off-the-Shelf Equipment
* ATCS Subsystems Simulation
* Control Flow Validation: Static and Dynamic
* ATCS Subsystems Emulation
* System Interoperability Testing

THE NEED FOR AN ATCS SYSTEM INTEGRATION SIMULATOR/EMULATOR/TESTER

* Validating System Safety
* Integration Testing (Communications Protocol and Data Comm.)
* Verify/Develop Specifications
* Assist Subsystem Design Process
* Human Factors Evaluation/Definition
* Duplication and Resolution of Field Problems
* Software Verification Testing

THE NEED FOR AN ATCS SYSTEM INTEGRATION

SIMULATOR/EMULATOR/TESTER (cont'd)

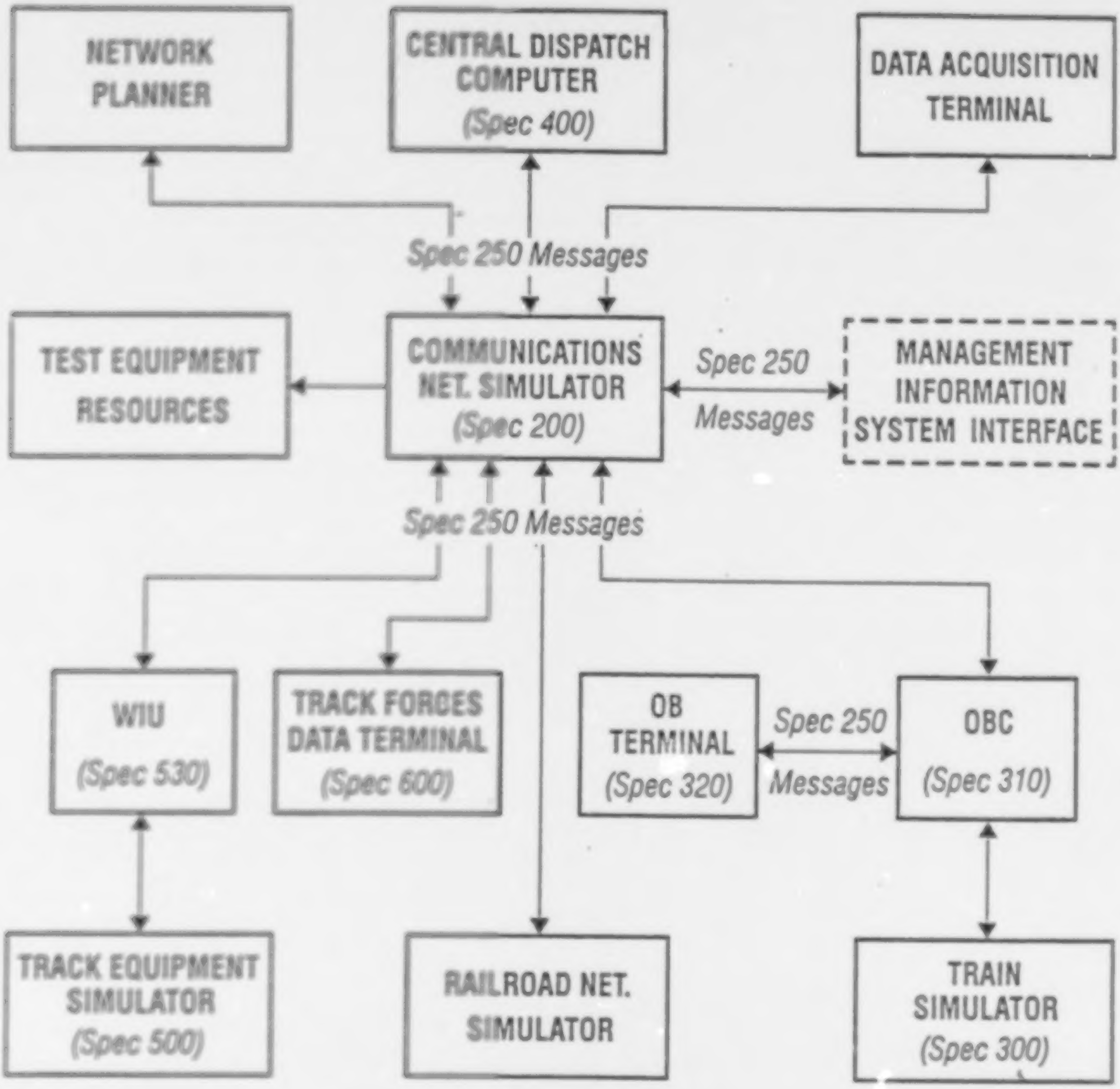
* Assisting Accident Investigations

* Helping Investigate New Regulatory Issues

* Develop Training Methods

* Develop Enhanced New Programs:

- Braking Algorithm
- Communication Procedures
- Operating Rules
- Etc...


NETWORK PLANNER
CENTRAL DISPATCH COMPUTER (Spec 400)
DATA ACQUISITION TERMINAL
Spec 250 Messages
TEST EQUIPMENT RESOURCES
COMMUNICATIONS NET. SIMULATOR (Spec 200)
Spec 250 Messages
MANAGEMENT INFORMATION SYSTEM INTERFACE
Spec 250 Messages
WIU (Spec 530)
TRACK FORCES DATA TERMINAL (Spec 600)
OB TERMINAL (Spec 320)
Spec 250 Messages
OBC (Spec 310)
TRACK EQUIPMENT SIMULATOR (Spec 500)
RAILROAD NET. SIMULATOR
TRAIN SIMULATOR (Spec 300)

ATCS SISET

Lab Architecture

ATCS SISET LAB ARCHITECTURE

* Data Communication Network:

- Dual Redundant Packet-switching Computers
- Base Station Computers
- Radio Transmitters/Receivers

ATCS SISET LAB ARCHITECTURE (cont'd)

* Locomotive On-Board Systems:

- More than one communicating sets of on-board equipment (OBC + MCP) (3) + on-board terminal
- One of them with full dynamic simulation capability

ATCS SISET LAB ARCHITECTURE (cont'd)

* Wayside Interface Units:
    - More than one communicating sets of WIU
* Central Dispatch Computer:
    - One with communication equipment
* Data Acquisition Terminal:
* Network Planner:
* Management Information System Interface:
    - With communication equipment

ATCS SISET LAB ARCHITECTURE (cont'd)

* Real-Time Dynamic Simulation Capability
    - Train Dynamics Simulator
    - Railroad Network Simulator
* Test Equipment Resources:
    - Data Message Analyzers
    - Automatic Data Logging
    - Protocol Analyzers
* Track Equipment Simulator:
* Track Forces Data Terminal:

EXPECTED COST

$5 Million Canadian

Over 3 Years.

CONCLUSIONS

* The Development of an ATCS System Safety Validation Program is a must.

* Interoperability and Conformability are the Cornerstones for Insuring ATCS System Safety.

* Verification and Validation of Software and Communications are necessary for a Safe Operation Throughout ATCS Life Cycle.

RECOMMENDATIONS

- The Development of the ATCS System Integration Simulator/Emulator/Tester Laboratory is recommended.

- Issues to be resolved:
    - Ownership
    - Location
    - Operation
    - Development

RECOMMENDED GOVERNMENT OBJECTIVE

It is recommended that one R&D objective of Government be to promote and support the development and deployment of a SISET Laboratory in Canada.

JUSTIFICATION

Government financial participation is justified in view of the Laboratory's potential to:

- Perform safety related functions throughout the ATCS life cycle; and
- Contribute to enhancing Canada's Technological and Manufacturing capabilities.

POTENTIAL TECHNOLOGICAL AND MANUFACTURING CONTRIBUTION

* Maintain Canada's Lead in ATCS Related Technologies:
    - Communications
    - Computer Software
    - Real Time Systems
    - Discrete Event Simulation
    - Vehicle Dynamics and Control
* Enhance Canada's ATCS Related Manufacturing Capabilities:
    - DSL
    - ALCATEL Canada/SEL Division
    - Motorola
    - Vapour Canada Inc.
    - Union Switch & Signal
    - Etc...

INDUSTRY DEMONSTRATED NEEDS

* Railways:

- Interested
- Skeptical about government interference
- Fear delays

* Suppliers:

- Very interested
- Anxious to cooperate

POTENTIAL PARTICIPATING GOVERNMENT ORGANIZATIONS

* Transport Canada

* Industry, Science and Technology

* Department of Communications

* National Research Council

RECOMMENDED GOVERNMENT SUPPORT

* Government should promote the establishment of a SISET Lab in Canada by the Railways, their Associations & Supplier Industry

* Industry should have the primary role & responsibility for such a facility

* Government funding participation in developmental costs could be for SISET'S safety related functions

* Government participation in annual operating costs could be through provision of some facilities to house and operate lab

APPENDIX B

List of Contacts

APPENDIX B

List of Contacts

George Achakji	Transport Canada	(613)990-7748
Roy Allen	Transportation Test Center	(719)584-0501
Ino Attas	VIA Rail	(514)871-6567
Robert Ayers	ARINC	(301)266-4741
Jack Bailey	ARINC	(301)266-4059
Robert Ballantyne	RAC	(514)849-4274
Mark Barbarash	VIA Rail	(514)871-6680
Pierre Berthiaune	CN Technical Center	(514)399-6017
John Brohm	Alcatel (SEL)	(416)445-8600
Roger Burns	Rockwell International	(319)395-4563
W.C. Chandler	Rockwell International	(319)395-1000
John Coleman	NRC	(613)998-9638
Peter Detmold	ATCS	(514)849-6913
Steven Ditmeyer	BN	(913)661-4168
David Ferguson	BN	(612)782-3108
Walter Friezen	Alcatel (SEL)	(416)445-8600
Roland Gagne	NRC	(613)993-2030
Bill Gallagher	CN	(514)399-8064
W.E. Goodman	FRA	(202)366-2231
John Granath	IITRI	(312)567-4450

Kelly Gravelle	Vapor Canada	(514)335-4200
James Gusso	CN Technical Center	(514)399-4252
Philip Hamill	NRC	(613)993-9448
Connie Hatherington	BN	(913)661-4173
Bill Hayes	NRC	(613)993-9443
R.N. Hodges	Freightmaster	(817)293-4220
M.H. Kelley	DSL	(514)744-5571
Robert Krick	FRA	(202)366-0453
Howard Moody	AAR	(202)639-2202
Bill Moore Ede	CN	(514)399-4057
R.L. Nash	CP Rail	(514)395-7445
Bill Paxton	FRA	(202)366-0425
Charles Radgowski	IITRI	(312)567-4459
Walt Rockey	FRA	(202)366-0897
Alan Rumsey	Alcatel (SEL)	(416)445-8600
Lu Sanders	ARINC	(301)266-4685
Andrew Schiestl	BN	(913)661-4164
Bob Shea	CP Rail	(514)395-6725
Gerry Spence	CN	(514)399-4328
Shafter Stotts	FRA	(202)366-4094
Garold Thomas	FRA	(202)366-0468
Helen Tucker	NRC	(613)998-9639
Bob Vadas	VIA Rail	(514)871-6000

J.W. Walsh	FRA	(202)366-0488
Frank Wilson	Rockwell International	(319)395-4552
John Wilson	DSL	(514)744-5571

APPENDIX C

Draft British Standard on
Functional Safety of Programmable Electronic Systems

BSI
STANDARDS

2 Park Street
London W1A 2BS
Telephone: 01-629 9000
Telex: 266933 BSILON G
Fax Gr 2/3: 01-629 0506

Document: 89/33005 DC

Our ref PCL/1/1

Date November 1989

DEADLINE FOR COMMENTS ON THIS DOCUMENT HAS NOW PASSED.
THE IEC/WG DEALING WITH THIS DRAFT IS NOW MODIFYING IT IN ACCORDANCE WITH THE COMMENTS RECEIVED.

DRAFT FOR PUBLIC COMMENT

Sub-committee PCL/1/1
System considerations
(of Technical Committee PCL/1
Measurement and control)

DRAFT BRITISH STANDARD 0000 : FUNCTIONAL SAFETY OF PROGRAMMABLE ELECTRONIC SYSTEMS : GENERIC ASPECTS
PART 1 : GENERAL REQUIREMENTS

NATIONAL FOREWORD

This draft British Standard has been prepared under the direction of the Industrial Process Measurement and Control Standards Committee.

It is identical with document:

65A(Secretariat)96 : Draft. Functional safety of programmable electronic systems : Generic aspects. Part 1 : General requirements.

TERMINOLOGY AND CONVENTIONS

Because the text of a draft International Standard has been copied some terminology and certain conventions are not identical with those used in British Standards.

Compliance with a British Standard does not of itself confer immunity from legal obligations.

NOTE. This draft Standard is identical with a document issued by the International Electrotechnical Commission and is based on International discussion in which the UK has taken an active part. Technical comments on this draft will assist in the preparation of the International Standard. Subject to committee approval it is expected that the completed International Standard will be implemented in the UK by the publication of an identical (dual-numbered) British Standard. The draft National Foreword reflects this expectation, but it is not intended to imply that the text of 65A(Secretariat)96 is necessarily acceptable to UK as it stands.

There is a high probability that this text could be adopted by CENELEC as a reference document for harmonization or as a European Standard. Recipients of this draft are requested to comment on the text bearing in mind this possibility.

It is recommended that in commenting on this document, National Committees should take account of the draft for "Software for computers in the application of industrial safety-related systems" prepared by Working Group 9, and for which a Secretariat draft has been issued (65A(Secretariat)94).

CONTENTS

Page

Introduction 4

Clause:

1. Scope 6
2. Normative references 9
3. Definitions and explanations of terms 10
4. Systems considered 21
5. Safety lifecycle model 23
6. Generic safety principles: System safety elements 41
7. Assessment general framework 46
8. Personnel competency 49
9. Safety plan 50

Annexes

A: Risk and safety integrity 59
B: Considerations underlying the guidance 70
C: Examples of different PESs to illustrate embedded and application software 77
D: System lifecycle model 79
E: Case study to illustrate aspects of the safety lifecycle 82
F: Reserved for future case study 88
G: General documentation requirements 89
H: Abbreviations used in the document 102
I: Bibliography 103

INTRODUCTION

Computer based systems, generically referred to as Programmable Electronic Systems (PES's) are increasingly being used for process and machinery control applications. This has created a demand for their use in safety related applications. If this new technology is to be effectively exploited it is essential that those responsible for making decisions in this area have sufficient guidance on the safety aspects on which to base those decisions. A key element in the application of PES's for safety related applications is the assessment of the safety integrity that has been achieved in a particular design. It is recognised that there is a great variety of PES applications covering a wide range of complexity, risk and hazard potential. The exact prescription of safety measures will therefore be dependent upon the specific application. This generically based International Standard will enable such a prescription to be formulated in future application-specific International Standards.

This International Standard has been developed:-

- To enable the functional safety of PESs used in safety related applications to be systematically addressed. This will provide a basis for carrying out the many activities associated with the Safety Lifecycle of safety-related systems - including design and assessment.
- With a rapidly developing technology in mind. Since both PES and the applications to which PESs are put, are developing, it is essential that the guidelines framework is sufficiently robust and comprehensive to cater for future developments.
- To allow future development of the Standard to reflect changes in safety integrity criteria and assessment techniques.
- To allow future application specific International Standards, dealing with the design and assessment of safety related PESs, to be developed.

A key part of this International Standard is the Safety Lifecycle Model (Cla 5). The Model has universal applicability and can be applied to configurat of safety-related systems irrespective of the technology on which each saf related system is based. For example, one or more safety-related system may based on hydraulics, pneumatics, electromechanical relay logic, electronic programmable electronics etc.

This International Standard is Part 1 of series dealing with the functi safety of programmable electronic systems. It is proposed that other P provide more detailed guidelines that will include:

- Hazard analysis.
- Risk assessment.
- System safety elements.
- Safety integrity requirements.
- System integrity levels.
- System validation.
- Retro-fitting.
- Documentation.

The considerations underlying the guidance are given in Annex A.

FUNCTIONAL SAFETY OF PROGRAMMABLE ELECTRONIC SYSTEMS: GENERIC ASPECTS

1. SCOPE

This International Standard provides guidelines on the aspects that need to be addressed when programmable electronic systems (PESs) are used to carry out safety functions. In particular it:

- applies to safety-related systems when one or more of such systems incorporate programmable electronics or other complex electronic devices.

- is generically based and can be used for both design and assessment purposes irrespective of the application. Examples of the application sectors coming within the scope of the Standard include:

  - process industries (emergency-shutdown systems, fire and gas detection systems, boiler controls).

  - manufacturing industries (industrial robots, machine tools).

  - transportation (railway signalling, braking systems, lifts).

  - medicine (miscellaneous electro-medical apparatus, radiography)

- is mainly concerned with safety to persons but is also applicable to environmental issues.

- applies to the total configuration of safety-related systems (ie. the overall hierarchy of safety-related systems).

- uses a Safety Lifecycle Model for all activities necessary for ensuring that the required safety-integrity levels are met for the safety-related systems under consideration. The Safety Lifecycle Model can be used for safety related systems irrespective of the technology on which such systems are based eg electrical/electronic/programmable electronic/hydraulic/pneumatic.

- addresses all relevant aspects associated with the design assessment of safety related systems (coming within the Scope of document). That is, a systematic approach is adopted covering total system environment of which the safety related systems are part.

- has been developed to enable the responsible Technical Committees develop application-specific International Standards.

- provides guidance on the application of PESs having safety functions fields where no specific recommendations are available.

- introduces the concept of "safety integrity levels" for specifying level of safety integrity of safety-related systems. The system saf integrity level is specified taking into account the tolerable r level associated with equipment under control. This Internatic Standard established PES safety integrity requirements based on safety integrity level to be achieved, but does not provide guideli for classifying the safety integrity level required for a speci application. The appropriate safety integrity level for a speci application must be established based on knowledge of the applica sector or other related experience. It will be for the respons Technical Committees, dealing with specific application sectors. specify the safety integrity levels for associated risk levels.

Note: 1) The guidance in this International Standard is primarily conce with safety related systems incorporating programmable electronics (PESs other complex electronic devices which have a large number of operating st and where testing of all input and output combinations and internal seque cannot be carried out. However, the guidance encompasses all types of sys based on electrical, electronic or programmable electronics. All the guidanc applicable to PESs whereas selective use of the guidance should be made for o electrical/electronic systems.

Note: 2) Although this International Standard is mainly concerned with safety to persons (but is also applicable to environmental issues) and concentrates on safety-related systems, it is recognised that the consequence of failure could also have serious economic implications. In such cases the document may also be applicable when assessing the protection of equipment and product.

2. NORMATIVE REFERENCES

The following standards contain provisions which, through reference in this tex constitute provisions of this International Standard. At the time publication, the editions indicated were valid. All standards are subject revision, and parties to agreements based on this International Standard a encouraged to investigate the possibility of applying the most recent editions the standards indicated below. Members of IEC and ISO maintain registers currently valid International Standards.

- ISO 9001: 1987: Quality Systems - Model for quality assurance design, development, production, installation and servicing.
- IEC Draft: Software for computers in the application of industri safety-related systems (65A (Secretariat) 94).

3. DEFINITIONS AND EXPLANATION OF TERMS

The definitions and explanation of terms for the purpose of this publication are given in Clauses 3.1 - 3.28.

3.1 **Equipment Under Control (EUC):** equipment/machinery/apparatus/plant used for manufacturing or process activities or any other activities in which safety related systems are used to;

- prevent hazardous events from taking place or;
- mitigate the effects of the hazardous events.

3.2 **System:** A system is the total entity of elements whose boundary extends from sensors, or other input devices, on the EUC (Clause 3.1) to the actuators, or other output devices, on the EUC.

3.3 **Programmable Electronic System (PES):** a system based on one or more central processing units (CPUs), connected to sensors and/or actuators, for the purpose of control, protection or monitoring.

NOTE: 1) The term PES includes all elements in the system extending from sensors, or other input devices, via data highways or other communication paths, to the actuators, or other output devices.

NOTE: 2) The structure of a PES is shown in Figure 1(a). Figure 1(b) illustrates the way in which a PES is represented in this document, with the programmable electronics shown as a unit distinct from sensors and actuators on the equipment and their interfaces. Figure 1(c) illustrates a PES with two discrete units of programmable electronics. Figure 1(d) illustrates a PES with dual programmable electronics but with single sensors and actuators. Figure 1(e) illustrates a system based on conventional hardware.

NOTE: 3) The term programmable electronic systems (PES) is used to cove systems incorporating a wide range of programmable electronic devices (se Clause 3.3) and includes systems incorporating:

- *Microprocessors*
- *Programmable Controllers (PCs)*
- *Programmable Logic Controllers (PLCs)*
- *Other computer based devices*

3.4 Programmable Electronics (PE): those parts of a PES which are not sole dedicated to a particular sensor or actuator and which perform differe functions at different times under the control of software. A CPU will t contained within any programmable electronics unit.

NOTE: The term specifically includes those parts of the system whose effe on the PES in the event of failure cannot readily be predicted. Exclud are sensors and actuators (unless they themselves contain programmab electronics) and any parts of the hardware which have no purpose other th to interface particular devices to the computer and whose effect on the P in the event of failure can be predicted.

3.5 Safety-related system: a system that:

- implements, independently of any other system, the required safe functions necessary to achieve a safe state for the EUC or maintain a safe state for the EUC.
- achieves, on its own or with other safety-related systems, t necessary level of safety-integrity for the implementation of t required safety-functions.

NOTE: 1) The term safety-related system refers to those systems that in fact the designated safety-related systems that enable, independently any other systems, the tolerable risk level to be met. (See clause 4.4)

NOTE: 2) The failure of a safety-related system would be included in the events leading to the identified hazard or hazards. Although there may be other systems having safety functions, it is the safety related systems that have been designated to achieve, in their own right, the required level of safety.

NOTE: 3) Safety-related systems can broadly be divided into two types; control systems and protection systems. It should be noted however, that not all control systems will be classified as safety related systems.

NOTE: 4) Safety-related systems may be an integral part of the normal control system for the EUC or may interface with the EUC by sensors and/or actuators. That is, the required level of safety integrity may be achieved by implementing the safety functions in the main control system of the EUC (and possibly by additional separate and independent systems as well) or the safety functions may be implemented by separate and independent systems dedicated to safety (eg. protection systems).

NOTE: 5) A person could be part of a safety related system. For example; a person could receive information from a programmable electronics device and perform a safety task based on this information or, perform a safety task through a programmable electronics device. (See Annex E, Figures E1 and E2).

3.6 Total configuration of systems: All those systems that make up the hierarchy of systems for control, protection or monitoring of the EUC.

Note: The total configuration of systems includes both safety-related systems and all other systems (the non-safety related systems). The failure rate and failure modes of the non-safety related systems may need to be addressed in the safety assessment. For example, failure of such systems will be one source of demand for operation on the safety related systems (see Annex E; Figure E2).

3.7 Total configuration of safety related systems: All the safety related-systems which, together, achieve the necessary level of safety integrity.

NOTE: The total configuration of safety related systems will comprise . many cases both PES and non-programmable systems (see Annex E; Figure E2).

3.8 **Hazard:** A physical situation with a potential for human injury.

NOTE: The term includes danger to persons arising within a short ti scale (eg fire and explosion) and also those that have a long term effect a persons health (eg release of a toxic substance).

3.9 **Control and protection systems:** Systems can be broadly categorised ir control systems and protection systems.

Control system: a system which responds to input signals from the E and/or from an operator and generates output signals causing the EUC operate in the desired manner.

Protection system: a safety-related system designed to respond conditions on the EUC, which may be hazardous in themselves, or if no act were taken, could eventually give rise to a hazardous event, and to gener the correct outputs to mitigate the hazardous consequences or prevent hazardous event.

NOTE: 1) All protection systems are, by definition, safety-related syst whereas some control systems may be designated safety-related systems.

NOTE: 2) All control systems, by definition, operate by control technic whereas protection systems operate by both control and non-cont techniques.

NOTE: 3) In many cases the correct response of the protection system wc be to bring all processes or movements to a halt; in others the corr response may be more complex eg involving an ordered shutdown or start-up equipment. The correct response may depend on the state of the EUC.

NOTE: 4) A control system may provide protection against failures aberrations on the EUC by normal feedback control action and may a incorporate protection features such as alarms and shutdown functic These protection features cannot, however, always protect against *potentially hazardous EUC states brought about by failure of the control functions since the initiating fault may affect both the control and protection functions. For this and other reasons it is common to provide separate control and protection systems.*

3.10 Safety integrity: the likelihood of a safety-related system achieving its required safety functions under all the stated conditions within a stated period of time.

NOTE: 1) In determining safety integrity, all causes of failures (both systematic failures and random hardware failures) which lead to an unsafe state should be included; for example, hardware failures, software induced failures and failures due to electrical interference. Some of these types of failure, in particular random hardware failures, may be quantified using such measures as the failure rate in the dangerous mode of failure or the probability of the protection system failing to operate on demand. However, the safety integrity of a system also depends on many factors which cannot be accurately quantified but can only be considered qualitatively.

3.11 Safety-related hardware reliability: that aspect of the safety integrity relating to random hardware failures in a dangerous mode of failure of the safety related systems.

NOTE: 1) The term relates to failures in a dangerous mode. That is, those failures of a safety-related system that would impair its safety integrity. The two reliability parameters that are relevant in this context being the overall failure rate and the probability of failure to operate on demand. The former reliability parameter is used when it is necessary to maintain continuous control in order to maintain safety; the latter reliability parameter is used in the context of protection systems.

2) The term, as used for the purpose of this International Standard, is narrower in meaning than is commonly used, relating only to hardware failures in a dangerous mode of failure. In common usage the term encompasses failure in both a safe and dangerous mode.

3.12 Random hardware failures: failures which result from a variety degradation mechanisms in the hardware.

NOTE: 1) There are many degradation mechanisms occurring at diffe rates in different components and since manufacturing tolerances c components to fail due to these mechanisms after different times operation, failures of a total equipment comprising many components occu unpredictable (i.e. random) times.

NOTE: 2) A major distinguishing feature between random hardware fail and systematic failures (see Clause 3.13), is that system failure rates other appropriate measure), arising from random hardware failures, can predicted with reasonable accuracy but systematic failures, by their nature, cannot be accurately predicted. That is, system failure r arising from random hardware failures can be quantified with reason accuracy but those arising from systematic failures cannot be accura quantified.

3.13 Systematic failures: failures due to errors (including mistakes and act omissions) in design, construction or use of a system which cause it to under some particular combinations of inputs or under some partic environmental condition. Systematic failures could arise, at any part the safety lifecycle.

NOTE: 1) Examples of systematic failure include:

- *Systematic failures due to errors in the system require specification*
- *Systematic failures due to errors in the design, manufac installation, operation of the hardware.*
- *Systematic failures due to errors in the design, implementa etc of the software.*

NOTE: 2) A failure of a safety-related system which is not caused by r hardware failure is by definition a systematic failure.

NOTE: 3) See also Clause 3.12; Note (2).

3.14 **Non-programmable system:** a system based on non-programmable hardware ie, a system not based on programmable electronics (PE) and which can be exhaustively tested.

NOTE: Examples would include hardwired electrical or electronic systems, mechanical, hydraulic, or pneumatic systems etc. Also included would be passive features such as additional containment devices (eg spillage retention bunds) and self-acting devices such as bursting discs or pressure relief valves. Excluded from this definition are complex electronic systems which cannot be exhaustively tested.

3.15 **System safety elements:** Three defined system characteristics of the safety related systems. They are:

i) Configuration: the specific arrangement of the programmable electronics within a PES and the type of safety-related systems (ie PES and non-programmable) that make up the total configuration of safety-related systems.

ii) **Safety-related hardware reliability:** (see 3.12)

iii) Systematic integrity: the non-quantifiable, qualitative aspects of the safety integrity of the safety-related systems.

NOTE: 1) Systematic integrity cannot usually be quantified -(as distinct from safety related hardware reliability which usually can).

NOTE: 2) See also Clause 3.11; Note (2).

3.16 **Conventional safety integrity:** the level of safety integrity which has been achieved, in situations similar to those under consideration, by safety related non-programmable systems which are well established and have traditionally been accepted as good engineering practice.

3.17 **Safety integrity requirements specification:** the specification that contains the requirements of the safety-related systems in order to meet the specified Integrity Levels.

NOTE: ***The three system safety elements, configuration, safety related hardware reliability and systematic integrity, will always form part of the safety integrity requirements specification although the exact package will be dependent upon the Integrity Levels to be achieved (for both the individual and the total configuration of safety-related systems).***

3.18 Failure/Error/Fault/Mistake:

Failure: A system failure occurs when the delivered service deviates from the specified service. The specified service is stated in the service specification and is an agreed description of the expected service.

Error: An error is that part of the system state which is liable to lead to failure. A failure occurs because the system is erroneous.

Fault: The cause of an error is a fault (eg hardware defect, software defect) which resides, temporarily or permanently in the system.

Mistake: Human error or fault. A human action (in carrying out any system lifecycle activity) that may result in unintended system bebaviour (failure).

Note: ***The above definitions are based on IEV and IEEE definitions taking into account new developments in the field of fault tolerant computing (mainly system and software related considerations). The above definitions resolve a number of inconsistencies of the older definitions when applied to programmable electronic systems.***

From the above definitions it can be seen that 'fault, error, failure' are part of a chain; that is:

[..... failure) (fault ...... error ...... failure)]

3.19 Safe-state: The state of the EUC when the specified hazard no longer exists.

NOTE: In going from a potentially hazardous condition to the final safe-state the EUC may have to go through a number of intermediate safe-states. For some EUC the only safe-state is for the to be continuously controlled which may be for a short or an indefinite period of time.

3.20 **System lifecycle:** The period of time that starts when a system is conceived and ends when the system is no longer available for use.

NOTE: The system lifecycle model used in this International Standard is described in Annex D (see Figure D.1).

3.21 **Safety lifecycle:** The elements of a system lifecycle that are safety-related.

NOTE: 1) The safety lifecycle model used in this International Standard is described in Clause 5 (see Figure 3).

3.22 **Risk:** The combination of the frequency, or probability, and the consequence of a specified hazardous event.

NOTE: The concept of risk always has two elements; the frequency with which a hazard occurs and the consequences of the hazardous event.

3.23 **Risk assessment:** Assessment of the EUC and the safety-related systems to ensure that the achieved risk level is lower or equal to the tolerable risk level.

3.24 **Tolerable risk level:** The tolerable risk level is the maximum level of risk of a particular technical process or condition that is regarded as tolerable in the context of the circumstances in question.

NOTE: In essence the risk level, as defined, specifies the level of safety to be achieved for the hazardous consequences associated with the EUC.

3.25 **Software:** Intellectual creation comprising the programs, procedures, rules and any associated documentation pertaining to the operation of a data processing system. (ISO/2382/1;2nd Edition; 1984-10-1)

NOTE: Software is independent of the carrier used for transport. In this International Standard the PES Software is considered to consist of the systems embedded software and the applications software. The embedded software is that software which is generic to that range of PESs and fundamental to the proper functioning of the PES but is not necessarily applications oriented. The applications software, by definition, is software specific to the user application. (Annex C gives different examples of PESs to illustrate embedded and applications software).

3.26 Development and production tools: Tools used as part of the design or assessment process.

NOTE: Examples would include CADCAM systems, compilers, expert systems and host target systems. If such tools are used as part of the design or assessment of the safety-related systems they should themselves form part of the safety assessment process.

3.27 Quality: The totality of features and characteristics of a product, proces or service that bear on its ability to satisfy stated or implied needs (IS 8402: Quality Assurance - Vocabulary).

3.28 Integrity Level: One of "m" possible discrete safety integrity levels fo specifying the safety-integrity level required for:

- the total configuration of safety-related systems
- individual safety-related systems
- safety related software

The Integrity Level m has the highest level of safety integrity; Integrit Level 1 has the lowest.

NOTE: 1) The number, m, is still under consideration.

NOTE: 2) The concept of integrity levels is fully compatible with 65A (Secretariat) 95; - see Clause 2. This latter document has 5 integrity levels -one of which is a boundary level (level 5) which is for non safety-related applications. (See Annex A, Clause A4)

4. SYSTEMS CONSIDERED

4.1 General

This International Standard is primarily concerned with safety-rel systems when one or more of such systems incorporate programm electronics (PE) or other complex electronic devices (see Clause 1; 1).

4.2 Programmable Electronic System (PES)

The concept of a PES is shown in Figure 1 (see also Clauses 3.3 and 3 The programmable electronics (PE) form an integral part of a PES. There be several programmable electronic units in any PES. These may exist serial linking arrangement or may exist in a parallel arrangement Figure 1 (b) and (c) respectively).

4.3 Configuration of Systems

The arrangement of safety-related systems, and the type of such sy could comprise:

- A single PES acting alone.
- A number of PES's acting in combination.
- A single PES acting in combination with one or more non-program systems.
- A number of PES's acting in combination with one or more programmable systems.

Typical configurations are shown in Figure 2.

4.4 Safety-related systems

The safety-related systems are those systems that enable, independentl other systems the tolerable risk level (see Annex A Figure A.1) to be

There may be other systems that perform safety functions or have an impact on safety in some way but it is the safety-related systems which, by definition, provide the necessary risk reduction in their own right irrespective of whether other systems, which may provide some safety measure, perform correctly (see also Clause 3.5 and associated Notes).

The safety-related systems are those systems that have been designated as the ones upon which the required level of safety integrity is to be assured. The concept of the safety-related system defined in this manner is fundamental to this International Standard.

4.5 Classification of system configurations

System configurations are classified as follows:

- Total configuration of systems - see Clause 3.6 (ie the complete hierarchy of all systems).
- Total configuration of safety-related systems - see Clause 3.7; (ie the complete hierarchy of all safety-related systems.).
- Individual safety-related system.

Annex E illustrates the use of the above classification.

NOTE: 1) With respect to Annex E; Figure E2, the classification is as follows:

Systems 1, 2, 3 and 4 are the total configuration of systems

Systems 3 and 4 are the individual safety-related systems.

Systems 3 and 4 constitute the total configuration of safety-related systems.

5. SAFETY LIFECYCLE MODEL

5.1 General

The Safety Lifecycle Model that is used as the core framework for this International Standard is shown in Figure 3. The model is relevant to any issue that needs to be addressed when carrying out any activity that has a bearing on the safety-related-systems. It therefore includes all system-lifecycle activities (see Annex D) and assessment activities. The key features of the model, with respect to Figure 1, are explained in Clauses 5.2-5.5. Use of this model, in the context of Assessment, is outlined in Clause 7.

The Safety Lifecycle should be applied, in the first instance, to all the systems forming the total configuration of systems (ie those which are safety-related and others).

NOTE: 1) The Safety Lifecycle can be applied in those situations where all, some, or none of the systems, in the total configuration of systems, incorporate programmable electronics or other complex electronic arrangements (see Clause 1; Note 1).

NOTE: 2) 'Assessment' is not shown on the Safety Lifecycle Model for reasons of clarity. In fact, assessment will be an integral part of all the safety-lifecycle activities (see also Clause 7).

5.2 Hazard analysis and risk assessment

5.2.1 Hazard Analysis

The hazard analysis comprises:-

(i) the identification of the potential hazards of the EUC; and

(ii) the identification of the events leading to those hazards.

The hazard analysis is concerned with a systematic analysis of the hazard: associated with the EUC. It should identify the hazards together with the event: which could give rise to them. Included in the events, leading to a specifi hazard, will be the identification of all the protective features (those which are dependent upon control techniques and those which are not). That is, it is necessary, in the first instance to identify all the systems in the event train of a specific hazard ie. the total configuration of systems (see Clause 3.6 and Clause 4).

Whether a formal hazard analysis technique is adopted will depend upon the seriousness of the risk in question and the complexity of the systems involved. If the combination of events leading to a hazard, involving systems depending upon control techniques and those depending upon miscellaneous protective features, are not easy to identify then a hazard analysis using special techniques will need to be carried out. There are several formal hazard analysis techniques, each with particular strengths and weaknesses and fields of application. The most widely used techniques are:-

- Fault tree analysis (FTA): used when a particular hazard or undesirable event has been identified and it is necessary to determine the combination of failures and operating circumstances which could cause the hazard. The method uses top down or deductive reasoning. Starting with the hazard or top event the analyst asks "What failure or event or combination of failures and events would cause the top event?". These contributing events are represented on the fault tree and are connected to the top event by an OR gate denoting that any one of the events would cause the top event, or by an AND gate denoting that the top event results only if all the events happen. The next stage is to consider how each of the contributing events could arise and to repeat the process until each event has been split into a logical combination of 'basic events' connected by logic AND or OR gates. (Other gates such as EXCLUSIVE OR, INHIBIT, PRIORITY AND and MAJORITY VOTE gates are also sometimes used). Basic events are those which do not require to be further developed since sufficient information exists on which to base the analysis eg. a frequency, failure rate or probability of failure is known for the 'basic event'.

  The completed fault tree is a logical representation of all the combinations of basic events which cause the top event. Each basic event may be treated as a Boolean variable and the logic for the top event can be written as a Boolean expression and manipulated and perhaps simplified by the laws of Boolean algebra. By su manipulation, or by inspection, those combinations of basic even which cause the top event, and which include a failure mode of the PE can be identified.

Fault tree analysis is a well accepted technique suitable for use identifying and representing combinations of events contributing to hazard. Its main disadvantages are that fault trees can become ve large and difficult to relate to the EUC and its operation and they c be difficult to quantify. Their accuracy relies on the ability of t analyst to deduce what can cause an event. (See Annex 1; Reference

- Failure mode and effects analysis (FMEA): used to determine the eff on the EUC of each mode of failure of each part of the system. reasoning used is bottom up or inductive. The analyst asks "W happens if this component fails in this particular failure mode?". general, it can be applied at any level within the system subsystem, module or component level according to the resour available and the level of detail required. (See Annex 1, Refere 2)

- Failure mode, effect and criticality analysis (FMECA): an extensior FMEA were each part failure is ranked according to the seriousness its effect and its probability of frequency of occurrence. several levels of hazard are possible then a FMECA is usefu determining the most critical components and in identifying wh reliability engineering resources should be allocated to grea effect.

- Hazard and operability studies (HAZOP): a form of FMEA which has developed for the chemical industries. The analyst studies instrumented flow diagram (sometimes called the piping and instru (P & I) line diagram) and considers the effect on the plan deviations in the normal parameters of the substances contained every pipe and vessel. As an aid to thoroughness, the analyst guide words to ensure that every possible deviation is conside Examples of guide words are:

- high flow
- low flow
- no flow
- reverse flow
- high temperature
- low temperature
- high pressure
- low pressure

The effects of each of these deviations is considered in each phase of operation including maintenance, commissioning, testing, start-up, shutdown and failure of services.

Hazard and operability studies are useful for EUCs with the potential for a serious accident but which are too complex for possible accident causes to be identified with any degree of certainty by any means other than a systematic search.

Hazard and operability studies could be particularly relevant to PES safety integrity assessment since PESs are commonly used for the control of parameters in large and complex chemical plants and many of the parameter deviations considered in the HAZOP study could be a direct result of a PES failure. (See Annex I; Reference 3)

- Event tree analysis (ETA): a method of representing and evaluating the possible sequences of events following from a failure in a part of a system. The technique uses the same bottom up reasoning as FMEA in that the assessor asks "What happens if this component fails?". An event tree can represent multiple and cascade failures and so is useful when there are several lines of defence between the initiating failure and the final hazard. It is also possible to introduce a time dimension into an event tree and so represent the effect of different system response times.

- Failure logic diagrams: a hybrid form of FTA and FMEA though not as rigorous as these two methods.

The hazard analysis should be fully documented and include both elements of the analysis; ie the identification of the hazards together with the event sequences leading to them.

5.2.2 Risk assessment

The risk assessment is intended to determine whether the risk associated with the design of the EUC, together with its safety-related systems, is lower or equal to the tolerable risk level (see Clause 3.24 and Annex A).

Depending upon whether the risk level is specified qualitatively or quantitatively will determine the type of assessment that is carried out. If the tolerable risk level is specified quantitatively then a quantitative risk assessment (QRA) would be carried out.

The risk assessment will need to be carried out taking into account all relevant information that is available at that point in time. The chief purposes of the risk assessment are to ensure, to a first order approximation (in either a qualitative and/or quantitative manner), that the underlying design parameters are soundly based and to establish that the design as proposed will meet the tolerable risk level. For example, it might be considered that the tolerable risk level can be met without the need for any safety-related control systems. The risk assessment should establish whether or not that is the case.

Ensuring that the tolerable risk level is achieved is likely to be an iterative process involving continuous refinements around the 'hazard analysis - risk assessment loop'. It will be necessary when the final risk assessment iteration is carried out to ensure that the information used for the hazard analysis is still relevant and this should be confirmed before finalising the ris assessment. Annex A gives an overview of the concepts of Risk and Safet Integrity. Figure 4 shows how the risk level influencing factors ar communicated to the Risk Assessment phase of the Safety Lifecycle.

Note: *It is important to differentiate between the terms 'risk assessment' an 'safety validation'. The former is concerned with an assessment of the ris associated with the EUC (and certain protective features) against the tolerabl risk level. The latter is concerned with ensuring that the safety-relate systems meet in all respects the Safety Requirements Specification (both th Functional Requirements Specification and the Safety Integrity Requirement Specification). (See Clause 5.3)*

5.3 Safety Requirements Specification

5.3.1 General

The Safety Requirements Specification is divided into the:-

(i) **Functional Requirements Specification**

(ii) **Safety Integrity Requirements Specification**

The Functional Requirements Specification, in this context, is the specification for all the functional safety requirements of the safety related systems. The specification is concerned with the functions that need to be carried out by the safety-related systems. It is not concerned with the degree of integrity required of the functions. That is, the Functional Specification is not concerned with the degree of certainty, or likelihood, that a specific safety function will be carried out but rather establishing and defining those that are necessary for ensuring safety.

The Safety Integrity Requirements Specification establishes the requirements for achieving the level of safety-integrity necessary in performing the safety functions.

For example, the Functional Requirements Specification may state that when the temperature reaches 'x', valve 'y' should open to allow water to enter the vessel. The Safety Integrity Requirements for the safety related system controlling the various valves etc is a set of requirements that if met will ensure that the safety related systems perform the functions required of them with the required probability of succeeding. **System failure will result if the functions that need to be carried out are not carried out for the circumstances specified because either the required safety functions have not been specified correctly or due to inadequate system performance in carrying out the specified safety functions.**

5.3.2 Functional Requirements Specification [UNDER CONSIDERATION]

5.3.3 Safety Integrity Requirements Specification

It is the formulation of the Safety Integrity Requirements Specification th ultimately determines the level of safety that is achieved (with respect to t safety-related systems and assuming the Functional Requirements Specification correct). Figure 4 shows how the Safety Integrity influencing factors a communicated to the Safety Requirements Specification in the Safety Lifecycle.

It is a pre-requisite for the design and assessment of a safety related syst that the required level of safety integrity is known and specified in terms th can be used for design and assessment purposes. The required level of safe integrity to be achieved by the safety-related systems is specified as Integrity Level. There is a specified Integrity Level for the to configuration of safety-related systems (translated from the tolerable risk le for the EUC) and a specified Integrity Level for each safety-related system (Annex A; Clause A5 and Figure A.4).

The Safety Integrity Requirements Specification specifies the requirements the safety-related systems, in order to meet the specified Integrity Levels, terms of the three System Safety Elements:

- Configuration
- Safety-Related Hardware Reliability
- Systematic Integrity

The relationship of the three system safety elements to the Safety Lifecycle shown in Figure 5. (The principles governing these elements and their effects system safety integrity are covered in Clause 6).

5.4 Designation of the safety related systems

The concept of a safety-related system and its designation is fundamental to t International Standard. The prevention of a specific hazard can be considere the combination of one or more of the following:-

i) Protective features not dependent upon control techniques. These come within the general classification of safety related systems

ii) Protective features which are dependent upon control techniques. These come within the general classification of safety-related systems but are more specifically referred to as safety-related control systems.

NOTE: The term 'safety-related control system' means that the safety-related system is utilising control techniques for its method of operation. The term encompasses safety-related 'control systems' and protection systems utilising control techniques (see Clause 3.8).

This International Standard is concerned primarily with (ii) above. However, the level of safety achieved is dependent upon all the various protective features. (ie (i) and (ii)). The way this is dealt with in this International Standard is to take account of all protective features when developing the Safety Requirements Specification (specifically the Safety Integrity Requirements Specification).

The various systems making up the total hierarchy of systems, are classified as follows:

- Total configuration of systems (see Clauses 3.6 and 4.5).
- Total configuration of safety-related systems (see Clauses 3.7 and 4.5).
- Safety-related system (see Clauses 3.5 and 4.4).

This concept is illustrated in Annex E; Figures E.1 and E.2.

There will be many situations when not all of the systems are required in order to meet the safety integrity level. It is the total configuration of safety-related systems that achieves the safety integrity level. Consequently, the correct identification and designation of all the safety related systems is crucial from both a design and assessment viewpoint.

5.5 Design and Implementation

5.5.1 General

The particular activities, relating to 'Design and implementation', that need to be addressed relevant to safety-related systems are specified in Clause 5.5.2.. This and all other design and implementation activities shall be carried out in accordance with an established quality system such as that described in ISO 9001.

5.5.2 Safety Verification

Verification is the comparison at each stage in the design and implementation process to determine that there is a faithful translation from one stage into the next. Specifically, the following verification steps must take place in the development of a safety related PES:

- The design of the safety related system(s) must be compared to the Safety Requirements Specification to ensure that all of the requirements, including both the Functional Safety Requirements and the Safety Integrity Requirements Specification, are adequately accomplished in the design.

- The implementation (final fabrication and software code) of the safety related system(s) must be compared to the design documents to ensure that the implementation is consistent with the design.

Safety Verification should proceed concurrently with the design and implementation process, so each step of development is verified for each element within the safety-related PES before progressing on to the next development phase for that PES element.

NOTE: In some systems, the safety functions may be embedded in larger parts of the overall system design, in others it is implemented in separate equipment. In either case, the Safety Verification must include specific and separate consideration of the safety-related elements of the system.

5.5.2.1 Safety Verification Plan

Concurrent with the initial design development, a Safety Verification Plan must be established. This plan should be coordinated with the Safety Validation Plan (Clause 5.5.1), to ensure that the Verification and Validation process provides a complete evaluation of the system implementation relative to the Safety Requirements Specification. The Safety Verification Plan should address the following elements:

- Selection of verification strategies (design reviews, walk-throughs, mathematical and systematic analyses, module and subsystem testing).

- Selection, specification and utilisation of test and simulation equipment.

- Management and execution of Safety Verification activities.

- Documentation (auditable records of all Safety Verification activities and decisions).

- Evaluation policies and procedures.

- Failure resolution policies and procedures.

5.5.2.2 Design Verification

The design verification has to establish whether or not the Safety Requirements Specification is completely satisfied by the system design. To accomplish this, the following design documents must be examined to assure completeness, consistency and compliance with the Safety Requirements Specification:

- System Design Specification;
- Software Design Specification(s);
- Hardware Design Specification(s).

The design, as expressed in the design specification, should furthermore be evaluated for feasibility, testability and maintainability.

All actions associated with the design verification should be documented in an auditable form. Any discrepancies between the design and the Safety Requirements Specification must be resolved according to the procedures established in the Safety Verification Plan.

5.5.2.3 Implementation Verification

The implementation verification has to establish whether or not the safety related portions of the system design are completely and correctly satisfied i the final system implementation. Implementation verification should cover al safety-related hardware assemblies and subsystems as well as safety-relate software and the integration of the hardware and software into the final syste configuration.

Implementation Verification should include testing of each element of the safety related system to ensure that it performs as specified by the applicable desig specification. Testing should cover functional aspects of each element, as wel as failure behaviour. Where applicable, elements should be tested with stimul beyond the intended design range to verify design margins. Where applicable testing should be performed to substantiate mathematical and systematic models For example, worst-case circuit analyses should be verified by circu performance testing and Failure Modes and Effects Analyses should be verified b systematic or random fault-injection testing.

5.5.2.4 Verification Documentation

All safety verification activities should be documented and all documents shou be retained in an auditable form. At the completion of the Safety Verificati process, a Safety Verification Report should be prepared summarizing t verification activities and results. The report should include:

- Summary of verification strategy;
- Description of Design Verification methods and procedures;
- All design deficiencies and associated resolutions.

5.6 Safety Validation

5.6.1 General

Safety Validation is the process of determining the level of conformance of the final operational system to the Safety Requirements Specification. Validation consists of executing tests and applying mathematical and systematic analyses.

The Safety Validation process should be accomplished by qualified persons other than those responsible for design and implementation of the PES. The validation process is executed in accordance with the Safety Validation Plan (clause 5.6.2).

5.6.2 Safety Validation Plan

The Safety Validation Plan must be developed concurrently with the Design and Implementation of the safety-related systems. The plan should precisely identify all of the tests and analyses required to adequately validate that all requirements, established in the Safety Requirements Specifications have been met. The plan should also establish management methods for executing the tests, reviewing results and implementing corrective actions. The Validation Planning should be done by personnel other than the design engineers responsible for implementing the PES.

The Safety Validation Plan should identify all tests and analyses required to adequately ensure that the PES implementation meets all elements of the Safety Requirements Specification including the Functional Safety Requirements and the Safety Integrity Requirements. The Plan should include tests for all phases of system operation, including:

- start-up
- steady state
- shut-down
- maintenance
- conceivable abnormal conditions

The Safety Validation Plan should address the following elements:

- Selection of Validation strategies (mathematical and systematic analyses, functional testing, failure mode testing, process simulation testing, in-situ testing).
- Selection, specification and utilization of test equipment and test software, including process simulation tools.
- Management and execution of Safety Validation activities.
- Documentation (auditable records of all Safety Validation Activities and decisions).
- Evaluation policies and procedures.
- Failure resolution policies and procedures.

5.6.3 Safety Validation Test Specification

As part of the Safety Validation process, it is a necessary requirement that testing of the safety-related systems be carried out. Such testing shall include functional testing, failure mode testing, process simulation testing and in-situ testing. The testing strategy shall be detailed in the Safety Validation Plan (see Clause 5.6.2) and the detailed test requirements shall be specified in the Safety Validation Test specification.

Safety Validation tests, which shall be included in the Test Specification, fall into two distinct categories:

- Functional Testing, whereby the system performance relative to the Functional Requirements is verified, and
- Failure Mode Testing, whereby the performance relative to the Safety Integrity Requirements is verified.

The relationship between the Safety Validation testing and the Safety Requirements Specification is shown in Figure 6 .

5.6.4 Safety Validation Analyses

As part of the validation process, particularly for validating the Safety Integrity Requirements Specification, it may be necessary to perform mathematical and systematic analyses. In Table 1, some appropriate analysis tools are listed for validation of each system element. These tools are listed as appropriate examples, other methods may be acceptable. The relationship between the Safety Validation Analysis and the Safety Requirements Specification is shown in Figure 6.

Table 1
Analysis Tools

System Safety Element	Analysis Tools
Configuration	- Failure Modes, Effects and - Criticality Analyses (FMECA); - Fault Tree Analysis (FTA)
Safety Related Hardware Reliability	- Failure rate prediction calculations; - Combinatorial models; - Markov models; - Monte Carlo simulations - Consideration of n-failures
Systematic Integrity	- Check lists to ensure coverage of important systematic failure causes - FMECA; - FTA - etc

5.6.5 Safety Validation Tests

Validation Tests are executed according to the pre-established Safety Validation Test Specification. The purpose of the validation testing is to demonstrate that the system is able to fulfil its specified function over all ranges of process conditions and environmental variations which it may experience in actual operation. In this, the use of a process simulator is very often necessary. In-situ testing is desirable, but is often limited due to equipment constraints.

The safety-related PES should be exercised through simulation of input signals present during normal operating conditions, anticipated operational occurrences and accident or other abnormal conditions requiring PES action. Input data should be chosen both within and at the limits of their valid domains, for transient modes as well as normal operation. Data should also be chosen out of their validity domains to verify system robustness.

Equipment used for validation testing must be calibrated. Special hardware and software tools should be subject to quality assurance controls. Such tools and equipment must be identified in the Safety Validation Report.

5.6.6 Safety Validation Report

At the conclusion of the validation process, a Safety Validation Report should be issued summarising the tests and analyses which were accomplished and the results. The report should specifically identify:

- version numbers of all items under test;
- personnel responsible for testing;
- all test equipment and simulation tools;
- any analysis or test failures which occurred and the ultimat resolution of the condition.

All results must be documented and retained in a form which is auditable b persons not directly involved in the validation. Records of any analysis or test failures and the ultimate resolution of the failure condition shall be kept in a auditable form.

5.7 Operation and maintenance

This phase of the Safety Lifecycle is concerned with the operation an maintenance phase. It is important to ensure that the features incorporated int the design and the assumptions made during the initial (or previous) safety integrity assessments are still valid for the existing circumstances. The key elements of this phase being:

- Identification of changes. This should cover the procedures responsible for the identification, and also examination, of the change procedures themselves. Changes may be regulatory or technically based; caused, for example, by changes in the safety regulatory system or by technical changes required to modify system performance or, possibly generated by performance feedback from the system itself.

- Safety implications of the identified change. If need be, the relevant parts of the initial (or previous) assessment will need to be re-done.

- Monitoring of the safety related system performance and comparison with predicted performance. Also to ensure that the assessment assumptions are valid.

5.8 System Modification [UNDER CONSIDERATION]

5.9 Decommissioning [UNDER CONSIDERATION]

5.10 Retro-fit

Retrofit is the process of replacing a functional component or subsystem with a new component or subsystem of a different design. Retrofitting may be prescribed for any of the following reasons:

- New component is more reliable that the old one;

- Old component can no longer be maintained due to obsolescence of technology;

- New component offers greater functionality or performance than old component;

- Other safety, performance or commercial reasons.

The following requirements apply to retrofitting a new component or sub-system:

i) The Functional Requirements Specification shall be met.

ii) The Safety Requirements Specification shall be met. That is, there shall be no reduction in the safety integrity level achieved.

iii) The guidelines in this International Standard shall be adhered to.

In order to ensure that (i), (ii) and (iii) are met, it may be necessary to undertake an analysis based on the Safety Lifecycle (Figure 3). That is, there may be a need to carry out a Risk Assessment, Hazards Analysis and examine the Safety Requirements Specification for compliance.

The process for selecting and applying retrofit components is represented graphically in Figure 7.

For any retrofit process, a Retrofit Integration and Validation Plan should be developed. The Plan should identify the following items in detail:

- Off-line validation of new component(s).
- Step-by-step integration procedures, including shut-down of old component(s) and start-up of new component(s). Often a phased switch-over can be accomplished by starting the new equipment running in a parallel "shadow" mode with the old equipment. The new equipment receives the same inputs as the old one, but does one provide any output to the process. Results of the two systems running side-by-side can be compared; when enough testing has been performed on the new system in this mode, control can be switched over to it.
- Retrofit management procedures, including evaluation policies and failure resolution policies.

6. GENERIC SAFETY PRINCIPLES: SYSTEM SAFETY ELEMENTS

6.1 General

The strategy adopted in this International Standard for dealing with dangerous mode failures, of the safety-related systems, is based on the three system safety elements (see Clause 3.15). The system safety elements are intended to provide adequate precautions against both random hardware and systematic failures (see Annex B). They are:

- Configuration;
- Safety-related hardware reliability and
- Systematic integrity.

The exact package of system safety elements will depend on the application and therefore the level of safety to be achieved.

The principles relate, and shall therefore be applied, to the total configuration of safety-related systems required to achieve an adequate level of safety integrity for the hazard in question. The total configuration will comprise, in many cases, both PES and non-PES safety related systems. The principles governing the three system safety elements are given in Clauses 6.1.1, 6.1.2 and 6.1.3.

NOTE: It is proposed to develop in Clauses 6.1.1, 6.1.2 and 6.1.3 the minimum technical requirements for each system safety element and to do this in the context of the safety-related system Integrity Level requirements. Detailed technical requirements for each system safety element will be given in future parts of this International Standard. In the context of safety-related software the requirements are given in IEC Draft 65A (Secretariat) 94 - see Clause 2.

6.1.1 Configuration

The configuration system element plays an important part, together with the systematic integrity system element, in tackling systematic failure causes. This International Standard specifies the minimum configuration rules to ensure that an appropriate level of fault tolerance, against systematic failure causes, is achieved by the overall structure of the safety-related systems. [FURTHER

In general, a final retrofit system carries with it the same requirements for safety development as the original system it is replacing; either the new equipment must have been developed with equivalent or better safety requirements and procedures, or the activities in the safety lifecycle shall be repeated for the retrofit solution.

ELABORATION REQUIRED: UNDER CONSIDERATION]

6.1.2 Safety-related hardware reliability

The safety-related hardware system element relates to the reliability with respect to random hardware failures in a dangerous mode of failure of the safety-related systems. There are several ways in which the requirements for this system safety element can be specified. The two principles ways being specifications based on qualitative criteria (see Clause 6.1.2.1) or specifications based on quantitative criteria .

6.1.2.1 Qualitative criteria

The requirements specification for the safety-related hardware system element, based on qualitative criteria, shall cover all hardware elements within a safety-related system, and the total configuration of safety-related systems. The specification could, for example, be based on:

- Safety performance maintained in the presence of n-failures (where n is specified in the safety integrity criteria).

- "Fail safe" behaviour - where the failure modes for the components are defined in the safety integrity criteria and safe behaviour of the system must be demonstrated in the presence of these failure modes.

NOTE: Assessment based on qualitative criteria will involve a qualitative appraisal of the safety related systems, using engineering judgement, of both sensors and actuators on the EUC and the programmable electronics. This could be carried out by means of a Failure Modes and Effects Analysis (FMEA) eg at component level or by the carrying out of specific tests. Either or both of which could be used to check conformity with a prescribed requirement.

6.1.2.2 Quantitative criteria; safety-related systems level:

The requirements specification for the safety-related hardware reliability system element, based on quantitative criteria at the safety-related systems level, shall cover all hardware elements within a safety-related system and the total configuration of safety-related systems. In the context of safety-related systems required to operate continuously to maintain safety, the reliability parameter of interest will be the mean time between failure (or failure rate). In the context of protection systems the relevant reliability parameter will be the probability of failure on demand. (The demand could arise for example from failure of the main control system. (See also Clause 3.9).

6.1.2.3 Quantitative criteria: EUC level:

The requirements specification for the safety-related hardware reliability system element, based on quantitative criteria at the EUC level, shall specify the hazardous frequency rate, deemed tolerable, for hazardous event being considered (or an equivalent statistical parameter). This will enable the reliability contribution of the safety-related systems to be determined.

NOTE: 1) The essential difference between Clauses 6.1.2.2 and 6.1.2.3 is that in Clause 6.1.2.2 the specification applies directly and specifically to the safety-related systems. In Clause 6.1.2.3 the specification applies directly to the EUC but by the application of reliability techniques allows the specific requirements for the safety-related systems to be developed.

NOTE: 2) Specifications based on quantitative criteria will normally be appropriate for those situations requiring a high level of hardware reliability, particularly in industries where it has been normal practice to use quantified safety measures.

6.1.3 Systematic integrity

The systematic integrity system element refers to non-quantifiable, qualitative aspects of the safety integrity of the safety-related systems. It relates to the precautions taken throughout all phases of the safety lifecycle to minimise systematic failures (eg against errors or omissions in the specification of the safety related systems and systematic failures from all other causes including software). It is the systematic system element and the configuration system element that are intended to tackle systematic failures. That is, all dangerous mode system failures that do not arise due to random hardware failures.

The level of systematic integrity to be achieved determines the likelihood of systematic failures. The key factor in determining the level of systematic integrity to be adopted being the Integrity Levels required for:

- the total configuration of safety-related systems.
- the individual safety-related system.
- the safety-related software.

Important pointers can also be taken from the procedural and engineering measures traditionally used to achieve conventional safety integrity (see Clause 3.16).

The approach to achieving an adequate level of systematic integrity is based upon:

- Implementation of appropriate technical, procedural and managerial measures for the safety-related systems throughout all phases of the safety lifecycle.

- Achievement of an adequate level of quality for all the safety-related systems (see Clause 6.1.3.1).

- Implementation of all phases of the safety lifecycle by competent persons (see Clause 6.1.3.2 and Clause 8).

- An Assessment of technical, procedural and managerial measures taken at all phases of the safety lifecycle of the safety related systems (see Clause 6.1.3.3).

6.1.3.1 Minimum level of quality

All phases of the safety lifecycle of the safety-related systems should be carried out in accordance with an established quality system such as that described in ISO 9001.

6.1.3.2 Quality of implementation.

The safety-related systems should be engineered by, or under the direction of, competent and experienced persons having an understanding of safety engineering principles. (see also Clause 8).

6.1.3.3 Qualitative assessment

The procedural and engineering aspects of the safety related systems, particularly the programmable electronics, should be carefully and systematically examined to ensure an appropriate level of systematic integrity. The assessment should cover all phases of the safety lifecycle. Clause 7 provides a framework for carrying out the assessment.

NOTE: One method of undertaking the assessment is to use checklists. The purpose of such checklists is not to lay down specific requirements but rather to stimulate critical appraisal. (See Annex I; References 4 and 5).

7 ASSESSMENT GENERAL FRAMEWORK [FURTHER ELABORATION REQUIRED; UNDER CONSIDERATION]

7.1 General

As an essential part of gaining assurance that the safety-related systems have met the Integrity Level requirements, it is necessary to carry out an assessment of all technical, procedural and managerial aspects associated with the safety lifecycle activities. The depth and coverage of the assessment, and the degree of independence of the assessor, will depend upon a number of factors. These include:

- the Integrity Levels for the total configuration and individual safety-related systems.
- the level of safety-related hardware reliability. Since the Integrity Level determines the likelihood of systematic failures and the level of safety-related hardware reliability determines the likelihood of random hardware failures, there is a need to ensure that the Integrity Level is matched to the level of safety-related hardware reliability. This means that the safety-related hardware reliability can be used as an indicator of the system Integrity Level requirements.
- the degree of experience. A significant history of satisfactory operation in an identical environment may prove the adequacy of certain aspects of the system such as standard software and immunity to electrical interference.
- net benefit. Comparing the estimated improvement in the overall safety integrity gained by complying with a specific requirement and the cost of complying with that requirement, priority should be given to those measures judged to give the greatest net benefit.
- the level of complexity of the safety-related systems.

An assessment framework, based on the Safety Lifecycle, is given in Clause 7.2.

7.2 KEY STEPS

The Key Steps in the assessment procedure (see Figure 8) are given below in Steps 1-7. If after applying Step 3 it can be established that a conventional safety integrity level (see Calsue 3.16) has been achieved by safety-related systems based on established guidelines then the safety integrity requirements specified in those guidelines should be followed in carrying out Steps 4, 5 and 6. It will need to be clearly established that the programmable electronics do not have any effect, with regard to safety, on these safety-related systems.

Step 1: **Hazard analysis and risk assessment.**

- Identify the potential hazards; and
- Identify the events leading to those hazards.
- Compare the actual risk with the specified risk level.

Step 2: Safety-requirements specification.

- Decide on the Integrity Levels for the safety-related systems.
- Decide on the safety-functions to be carried out.

Step 3: Designation of the safety related systems.

- Identify in the first instance, the total configuration of systems. It is then necessary to decide which of those systems are to be designated safety-related systems. (See Clause 5.4).

Step 4: Design and implementation.

- Assess the design and implementation requirements and subsequent execution.

Step 5: Safety validation assessment.

- Assess the validation requirements and subsequent execution.

Step 6: Operation and maintenance.

- Assess operation and maintenance requirements.

Step 7: Assessment Report.

- Produce Assessment Report.

8. PERSONAL COMPETENCY

Before starting a project involving safety-related systems, those responsible for the project shall ensure that all persons involved in project activities are competent to carry out the duties required of them. Personal competency means that hose undertaking the specified duties shall have adequate and relevant:-

- technical knowledge,
- experience; and
- ability to follow procedures

What is "adequate" will depend upon several factors (eg. technical complexity, relevant experience, competency of supervision).

Particular consideration should be given to personal selection, training and the degree of responsibility held. In those areas where competency levels need to be improved then training (including where necessary refresher training) should be implemented.

The specialised nature of safety-related systems can impede communication between those involved with the detailed technicalities of the project and others who have an important part to play, from a safety viewpoint, but may not be involved with such details. This should not prevent the proper monitoring of the project so as to ensure that appropriate management and technical procedures are adhered to. Competency is an essential requirement for all members of the project team.

Once the safety-related systems have been implemented it is important that appropriate levels of personal competency are established, and maintained, in order to ensure that the safety-related systems are operated and maintained in accordance with the assumptions used during the safety-integrity assessment.

9. SAFETY PLAN [UNDER CONSIDERATION]

(It is proposed to include here the deliverables required, including documentation requirements, for each activity in the Safety Lifecycle).

FIGURE 1: DIAGRAM TO ILLUSTRATE PES STRUCTURE AND NOTATION

Figure 1a: Basic PES Structure

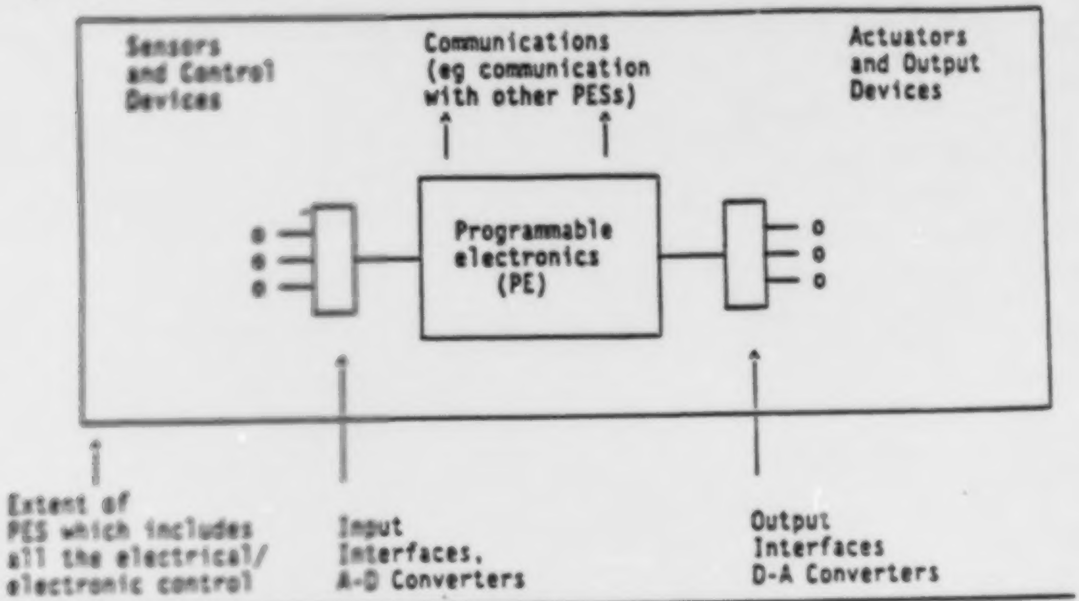
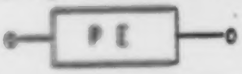

Sensors
and Control
Devices
Communications
(eg communication
with other PESs)
Actuators
and Output
Devices
Programmable
electronics
(PE)
Extent of
PES which includes
all the electrical/
electronic control
Input
Interfaces,
A-D Converters
Output
Interfaces
D-A Converters

Figure 1b: Diagrammatic representation of a single PES with single programmable electronics ie. one PES comprising one channel of programmable electronics.



PE = Programmable Electronics

Figure 1c: Diagrammatic representation of a single PES with dual programmable electronics linked in a serial manner (eg. intelligent sensor and programmable controller).

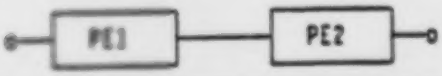


Figure 1d: Diagrammatic representation of a single PES with dual programmable electronics but shared sensors and actuators ie. one PES comprising two channels of programmable electronics.

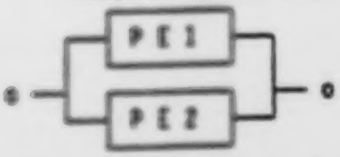
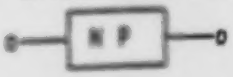


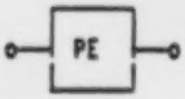
Figure 1e: Diagrammatic representation of a Conventional System, eg electrical/electronic/mechanical hydraulic/pneumatic etc.



NP = Non-programmable Hardware

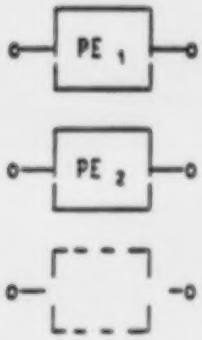
FIGURE 2: SYSTEMS UNDER CONSIDERATION: EXAMPLES OF VARIOUS CONFIGURATIONS (NOT EXHAUSTIVE)

Figure 2(a)



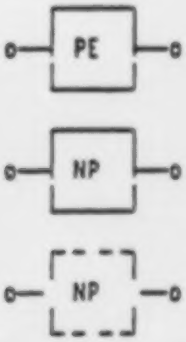
A single PES acting alone

Figure 2(b)



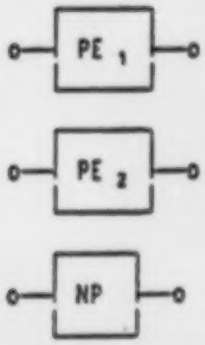
A number of PESs acting in combination

Figure 2(c)



A single PES acting in combination with one or more non-programmable system

Figure 2(d)



A number of PESs acting in combination with one or more non-programmable system.

Fig. 3 Safety Lifecycle Model

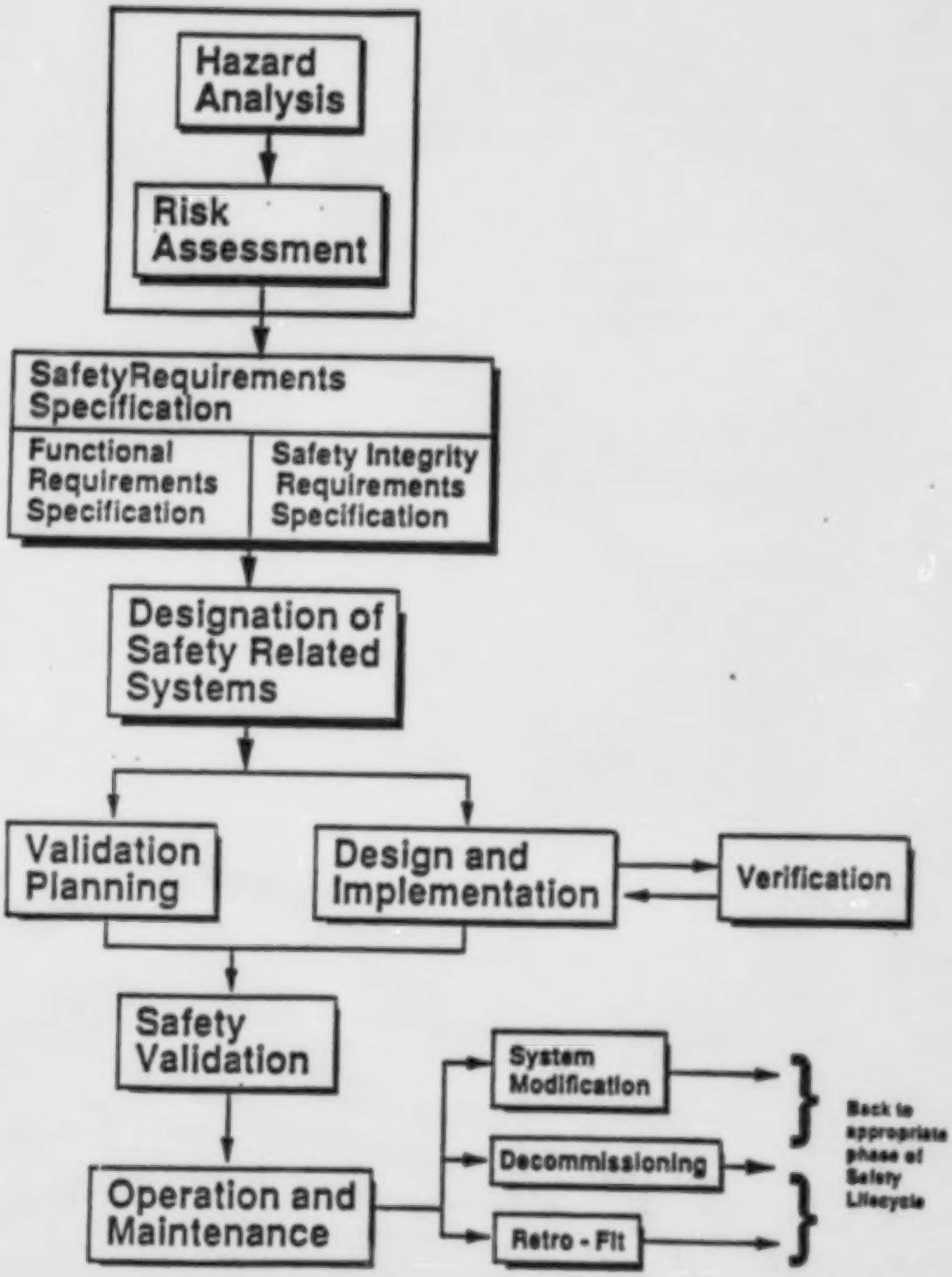

Hazard Analysis
Risk Assessment
SafetyRequirements Specification
Functional Requirements Specification
Safety Integrity Requirements Specification
Designation of Safety Related Systems
Validation Planning
Design and Implementation
Verification
Safety Validation
Operation and Maintenance
System Modification
Decommissioning
Retro - Fit
Back to appropriate phase of Safety Lifecycle

Fig. 4 Relationship of the risk and safety integrity levels to the Safety Lifecycle Model

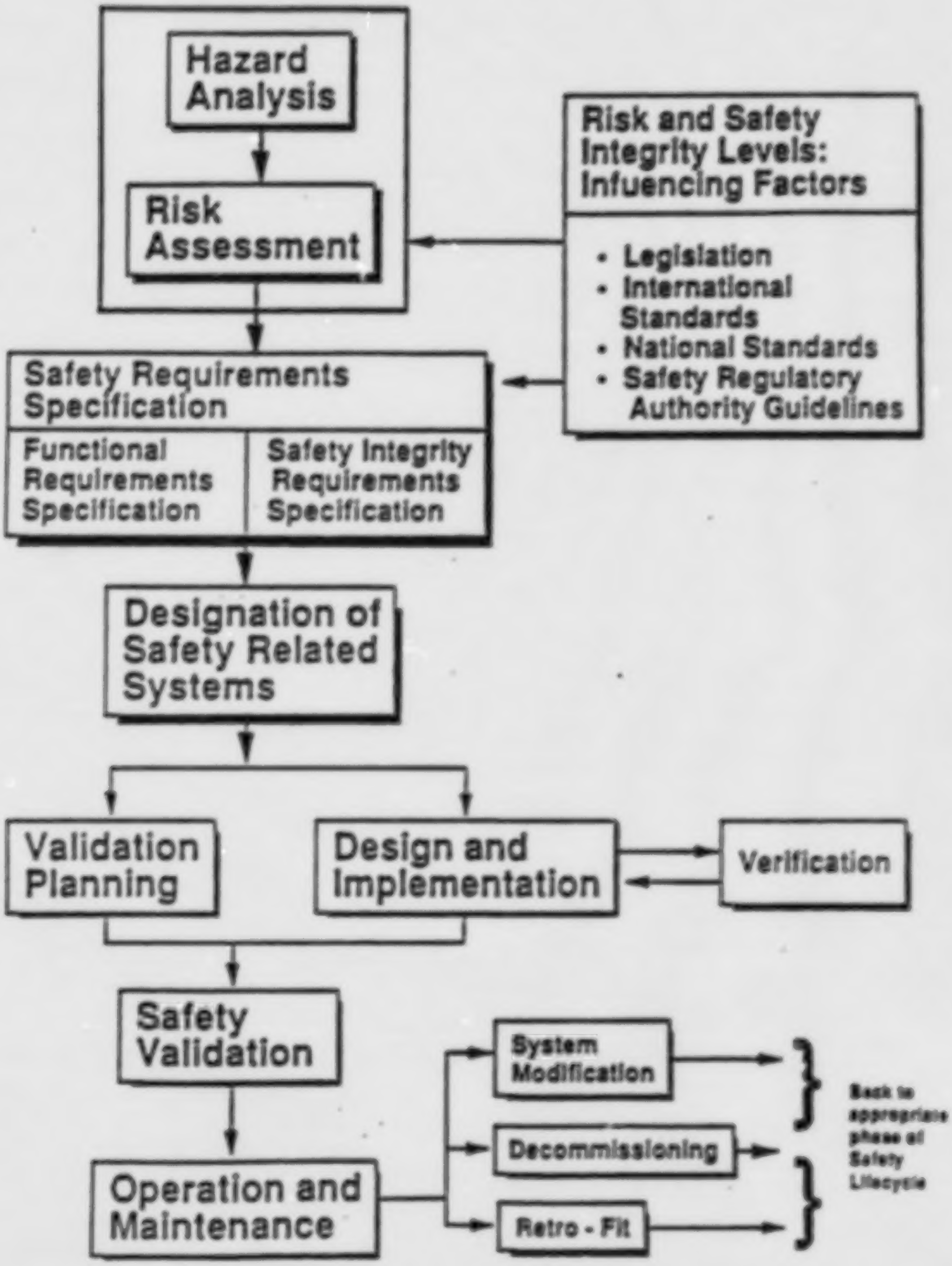

Hazard Analysis
Risk Assessment
Risk and Safety Integrity Levels: Infuencing Factors
• Legislation
• International Standards
• National Standards
• Safety Regulatory Authority Guidelines
Safety Requirements Specification
Functional Requirements Specification
Safety Integrity Requirements Specification
Designation of Safety Related Systems
Validation Planning
Design and Implementation
Verification
Safety Validation
Operation and Maintenance
System Modification
Decommissioning
Retro - Fit
Back to appropriate phase of Safety Lifecycle

Fig. 5 Relationship of the 3 system safety elements to the Safety Lifecycle Model

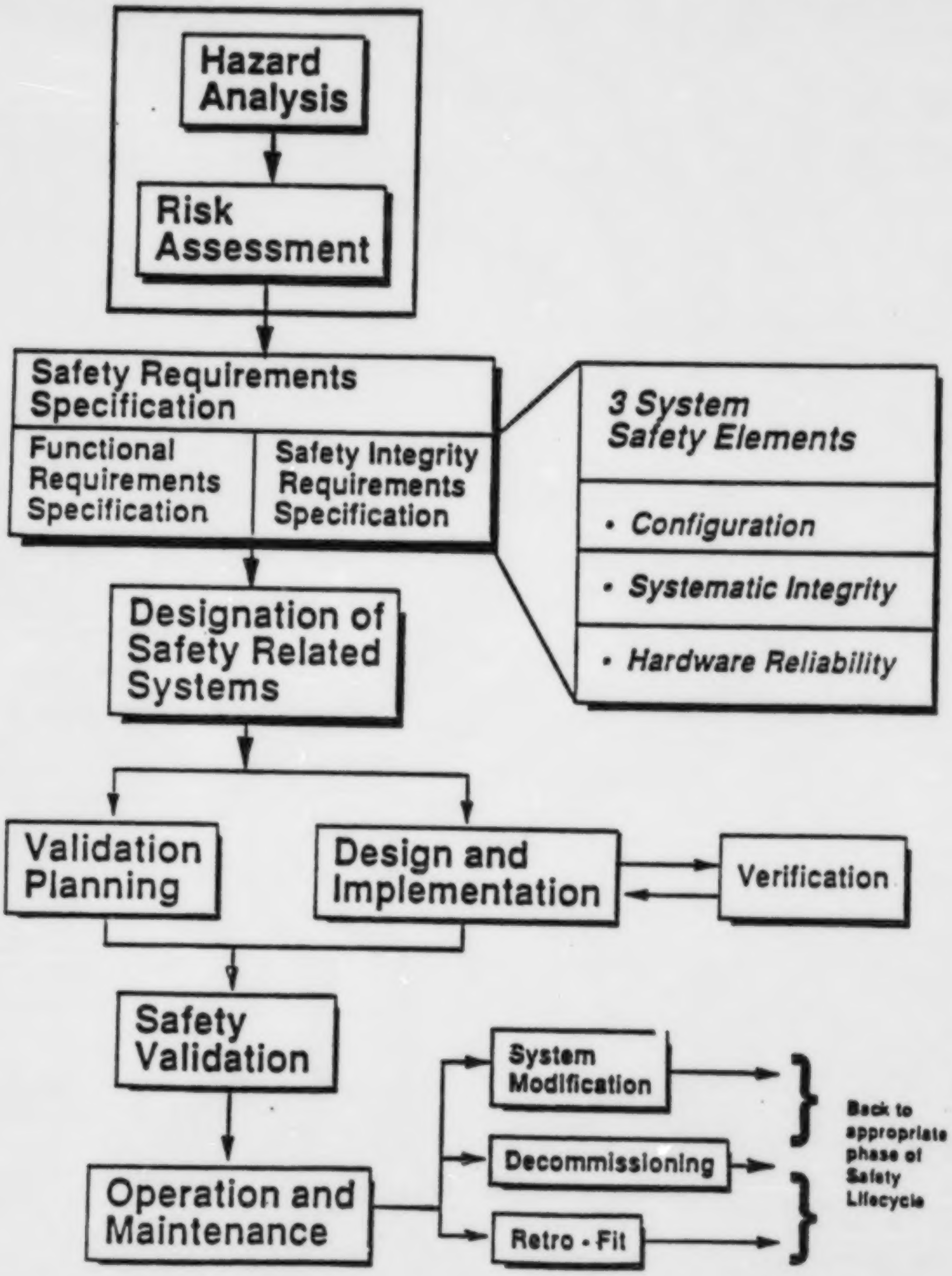

Hazard Analysis
Risk Assessment
Safety Requirements Specification
Functional Requirements Specification
Safety Integrity Requirements Specification
3 System Safety Elements
• Configuration
• Systematic Integrity
• Hardware Reliability
Designation of Safety Related Systems
Validation Planning
Design and Implementation
Verification
Safety Validation
Operation and Maintenance
System Modification
Decommissioning
Retro - Fit
Back to appropriate phase of Safety Lifecycle

Fig. 6 Safety validation analyses and testing: relationship to the safety requirements specification

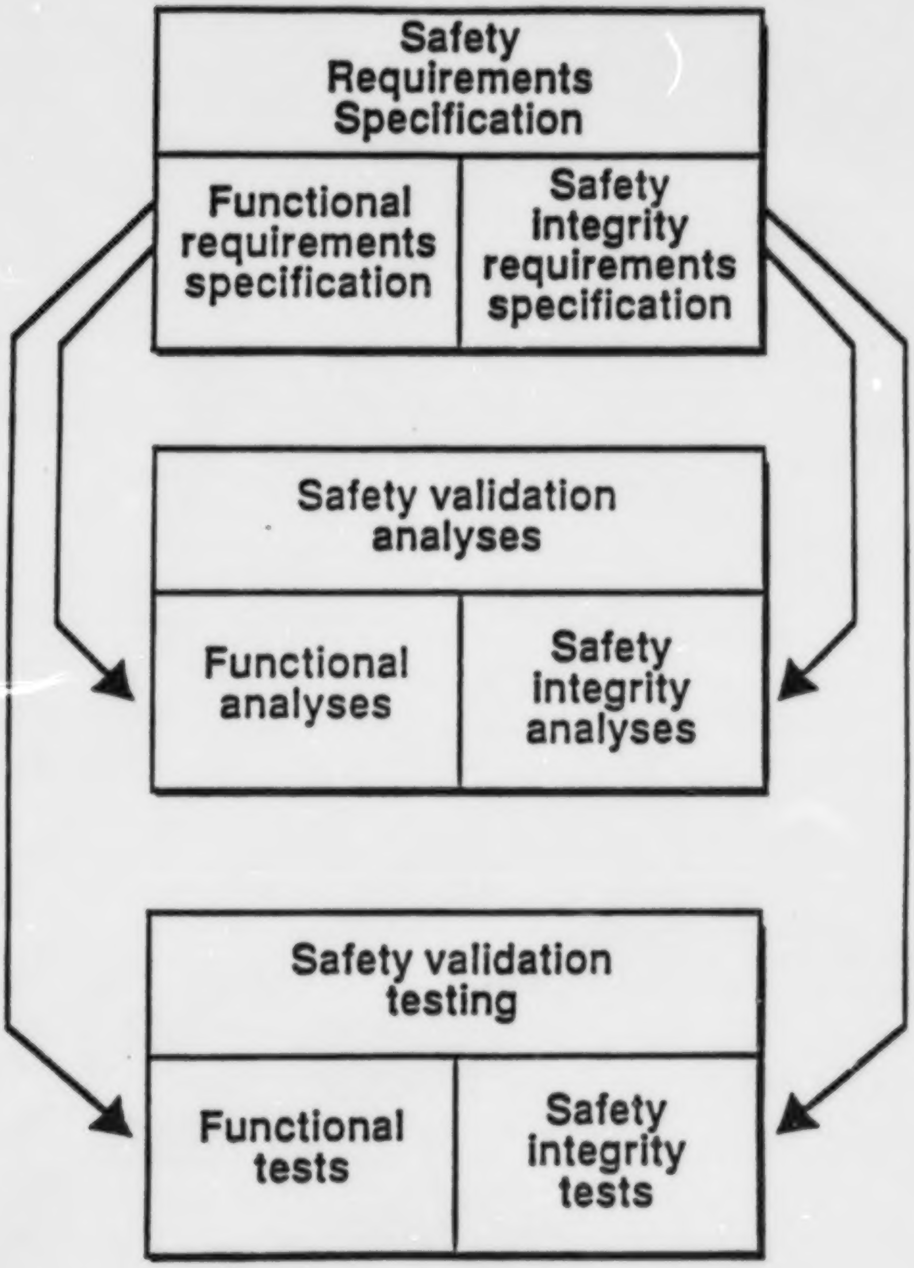

Safety Requirements Specification
Functional requirements specification
Safety integrity requirements specification
Safety validation analyses
Functional analyses
Safety integrity analyses
Safety validation testing
Functional tests
Safety integrity tests

Fig. 7

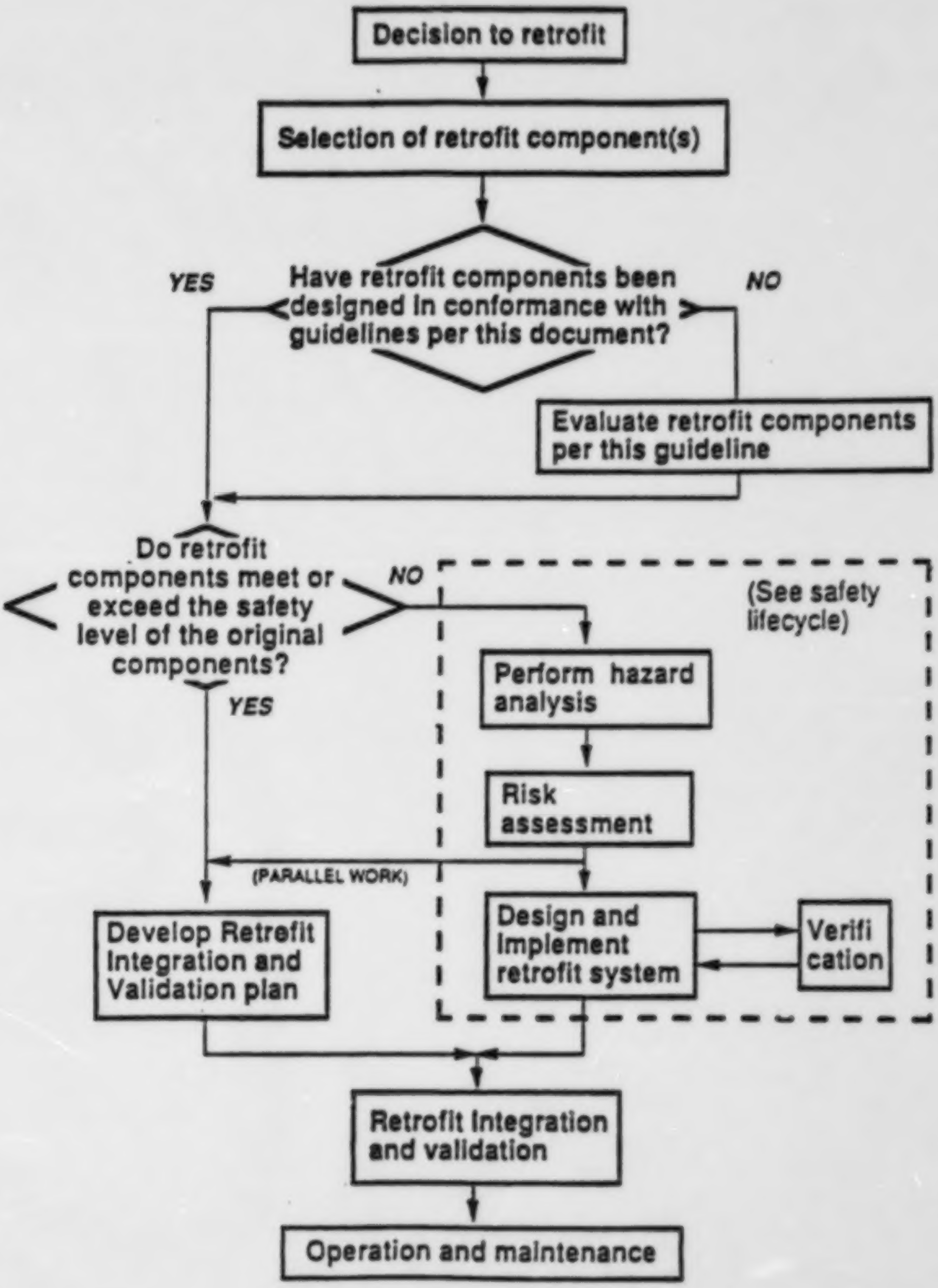
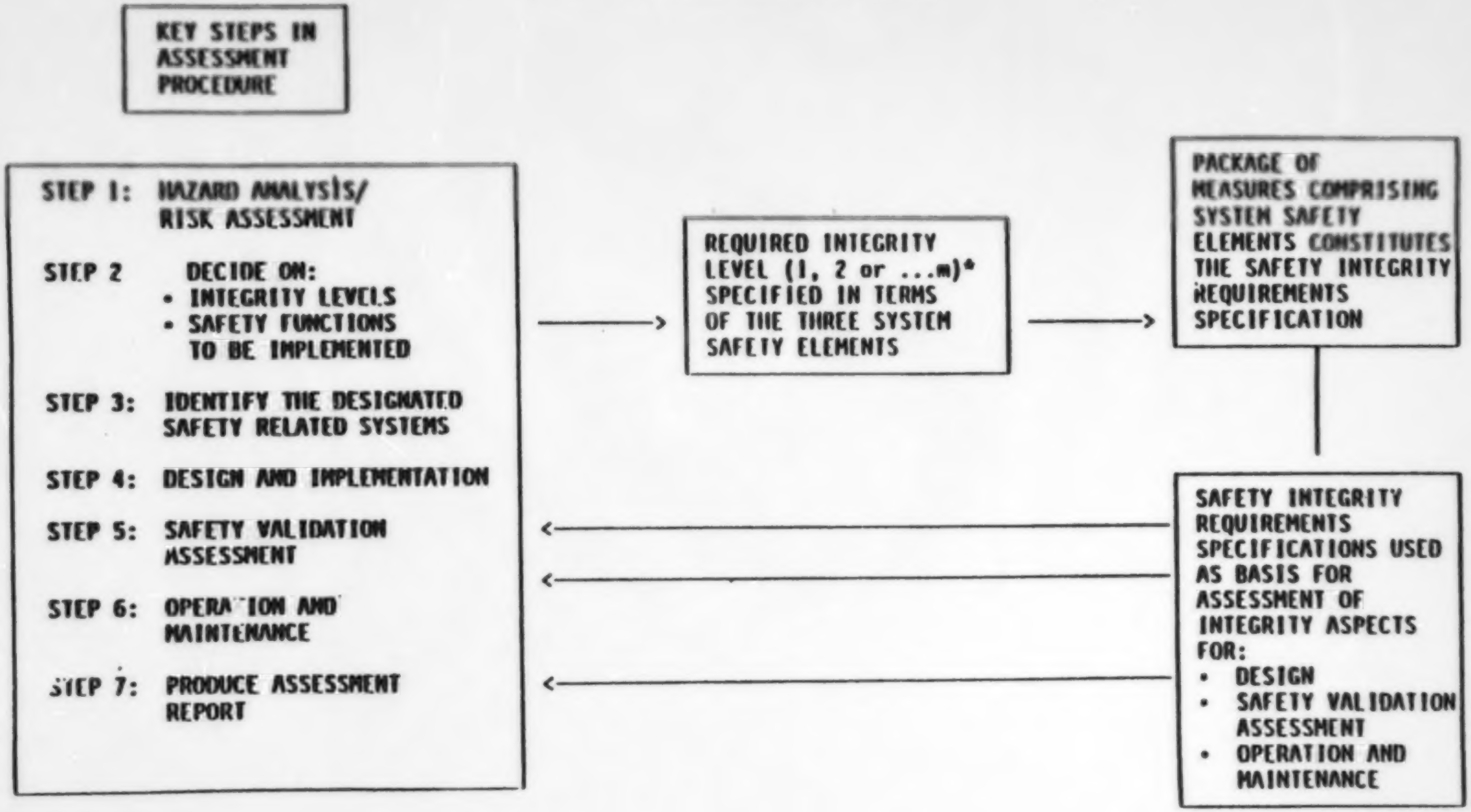

Decision to retrofit
Selection of retrofit component(s)
Have retrofit components been designed in conformance with guidelines per this document?
YES
NO
Evaluate retrofit components per this guideline
Do retrofit components meet or exceed the safety level of the original components?
NO
YES
(See safety lifecycle)
Perform hazard analysis
Risk assessment
(PARALLEL WORK)
Develop Retrefit Integration and Validation plan
Design and Implement retrofit system
Verifi cation
Retrofit Integration and validation
Operation and maintenance

FIGURE 8: ASSESSMENT FRAMEWORK


KEY STEPS IN ASSESSMENT PROCEDURE
STEP 1: HAZARD ANALYSIS/ RISK ASSESSMENT
STEP 2 DECIDE ON:
• INTEGRITY LEVELS
• SAFETY FUNCTIONS TO BE IMPLEMENTED
STEP 3: IDENTIFY THE DESIGNATED SAFETY RELATED SYSTEMS
STEP 4: DESIGN AND IMPLEMENTATION
STEP 5: SAFETY VALIDATION ASSESSMENT
STEP 6: OPERATION AND MAINTENANCE
STEP 7: PRODUCE ASSESSMENT REPORT
REQUIRED INTEGRITY LEVEL (1, 2 or ...m)* SPECIFIED IN TERMS OF THE THREE SYSTEM SAFETY ELEMENTS
PACKAGE OF MEASURES COMPRISING SYSTEM SAFETY ELEMENTS CONSTITUTES THE SAFETY INTEGRITY REQUIREMENTS SPECIFICATION
SAFETY INTEGRITY REQUIREMENTS SPECIFICATIONS USED AS BASIS FOR ASSESSMENT OF INTEGRITY ASPECTS FOR:
• DESIGN
• SAFETY VALIDATION ASSESSMENT
• OPERATION AND MAINTENANCE

* *(See Clauses 3.28 and A.4)*

ANNEX A (Normative)

RISK AND SAFETY INTEGRITY

A.1 Risk

Risk is defined as "The combination of the frequency, or probability, and the consequence of a specified hazardous event". The concept of risk always has two elements; the frequency, or probability, with which a hazard occurs and the consequences of the hazardous event.

In any particular application there will be a level of risk (the risk level) that can be deemed "tolerable". For situations where the level of risk, without precautions, is not tolerable, there is a need to ensure that the applied safety-related systems reduce the risk to the risk level. The general concept of risk is shown in Figure A. 1. It can be seen from Figure A.1 that the risk level is met, for example, by reducing the probability of the hazard occurring by adopting safety measures - including safety-related control systems (eg. a protection system).

By definition, risk is concerned with the frequency, or probability, of the hazardous event and the consequences of it. The tolerable risk or risk level to be achieved for a specific situation, which may be stated either qualitatively or quantitatively (eg. that the hazardous event, leading to specific consequences, shall not occur with a frequency greater than once in 10^4 years) will need to be translated into the safety integrity requirements for the safety-related systems in order that the design can proceed. This approach then allows an engineering solution to be adopted for the design of the of safety-related systems.

The purpose of the risk level is to state what is deemed tolerable with respec to both the frequency (or probability) of the hazardous event and th consequences of it. It is the EUC that creates the hazard. The safety-relate systems are designed to reduce the frequency (or probability) of the hazardou event and/or the consequences of the hazardous event.

A.2 Safety integrity

Safety Integrity is defined as "The likelihood of a safety related system achieving its required functions under all the stated conditions within a stated period of time". Safety integrity relates to the performance of the safety-related systems, from a safety viewpoint, and can be specified for the total configuration of safety-related systems or for an individual safety-related system.

The required safety integrity level, which is contained in the Safety Integrity Requirements Specification, is specified in terms of the three system safety elements (see Clause 3.15 and Figure 5). The required safety integrity of the total configuration of safety-related systems must be of such a level so as to ensure that the failure frequency of the safety-related systems does not allow the hazard frequency to exceed that stated in the tolerable risk level. Figure A.2 illustrates, by way of a simple example, the terminology for a total configuration of two safety-related systems.

A.3 Risk and safety-integrity separation

It is important that the reason for the separation of risk and safety-integrity be fully appreciated. Risk applies to the EUC; Safety Integrity applies to the safety-related systems. Tolerable risk levels are determined on a societal basis and involve consideration of societal and political factors. Tolerable risk levels are usually set by interested parties within the application sector in question. Determination of the tolerable risk level takes into account the specific hazardous consequences for the specific application. Safety-integrity is an engineering concept which allows the safety-related systems to be adequately specified, in engineering terms, for the tolerable risk level. Once the tolerable risk level has been set, the required safety integrity level for the total configuration of safety-related systems can be determined.

This separation of risk and safety-integrity is show in Figure A.3.

A.4 System and software integrity levels

The overall safety-integrity level for the safety-related systems is derived from the tolerable risk level for the EUC. From the overall safety-integrity level, which applies to the total-configuration of safety-related systems, the safety-integrity requirements for each individual safety-related system can be developed. Any software carrying out safety-functions (ie. the safety-related software) in one of those individual safety-related systems should have a level of integrity based upon the individual system safety integrity level, of which it forms a part, and the safety integrity level requirements of the total configuration of safety related systems (see Clause 6.1.1). That is, in any individual safety-related system, the safety-integrity of the software should be matched to the individual system safety integrity level. This concept of moving from a risk level to a software integrity level is shown in Figure A.4.

The transition from the tolerable risk level to the overall safety integrity level for the total configuration of safety-related systems, and then to the establishment of safety integrity requirements for individual safety-related systems, has important implications in the context of both system and safety-related software integrity levels. To cater for the very wide overall risk levels found in practical applications, there will be of necessity a wide spectrum of safety integrity levels for the total configuration and individual safety-related systems. It is useful, therefore, to have available a number of Integrity Levels for both safety-related systems and safety-related software. The Safety Integrity Requirements Specification, for the safety-related systems, is then developed based upon the Integrity Level specified for the total configuration of safety-related systems. The Safety Integrity Requirements Specification is specified in terms of the three system safety elements (see Clause 3.15 and 5.3). (The exact number of levels has not been finalised - see Clause 3.28)

In this International Standard "m" Integrity Levels are specified (see Note 1 below). Integrity Level 1 being the highest level; Integrity Level m being the lowest. The safety-related software in an individual safety-related system, of Integrity Level N, shall be required to be at least, by definition, of Integrity Level N. For example, if an individual safety-related system has an Integrity Level 2 designation, then the safety-related software in that system shall be of Integrity Level 2 (or Integrity Level 1) in order to ensure that the software does not reduce the system integrity of which it forms a part. The software integrity level may have to be above the minimum level in order to meet the requirements for the configuration safety system element (see Clause 6.1.1).

Note: 1) The exact number of Integrity Levels is still under consideration - see Clause 3.2.8.

Note: 2) The software requirements for particular software integrity levels are specified in IEC Draft 65A (Secretariat) 94; - see Clause 2.

Note: 3) See Annex I References 6, 7, 8, 9 and 10.

A.5 Risk to safety integrity transition

As indicated in Clause A.3, there is a necessary separation between Risk (of the EUC) and Safety Integrity (of the safety-related systems). However, when a tolerable risk level has been specified there is a need to translate this into engineering terms for input to the Safety Requirements Specification for the overall system (that is, the total configuration of safety-related systems). The tolerable risk level applying to the EUC is translated into one of m integrity Levels for the total configuration of safety-related systems. From this overall Integrity Level requirement, the Integrity Level requirements for each safety-related system is determined (see Figure A.4).

In determining the overall Integrity Level, from the tolerable risk level, it is necessary to evaluate the contribution the safety-related systems are making to the risk reduction. The safety-related systems may:

i) be designed to prevent, absolutely, the consequences of the hazardous event, ie. if the safety-related systems perform their safety-functions then no hazard arises. The key factor here is the ensuring that the safety-related systems perform their functions with the degree of certainty required (eg. that the probability of failure on demand should not be greater than 1 in 10^4 years).

ii) be designed to mitigate the effects of the hazardous event. That is, of the hazardous event - thereby reducing the risk by reducing the consequence (again, as for (i) the probability of failure in demand (or other appropriate statistical measure) should be met.

iii) be designed to achieve a combination of (i) and (ii).
Taking into account (i), (ii) and (iii), it is possible to determine, from the risk level, the Integrity Level for the total configuration of safety-related systems.

A.6 Risk and safety integrity level determination

It is a prerequisite prior to the design and assessment of safety-related systems that the overall Integrity Level is known and specified in terms that can be used for design and assessment purposes. Tolerable risk levels will depend upon many factors eg. severity of injury, number of people exposed to danger, frequency at which a person or people are exposed to danger and the duration of the exposure. Important factors will be the perception and views of those exposed to the hazardous event. Influencing factors in the development of risk and safety integrity levels include:

- Legislation
- International standards
- National standards
- Industry standards
- Safety Regulatory authority guidelines

The safety integrity level required in any specific application depends upon a number of complex factors - including the level that has traditionally been accepted as constituting good safety practice for the application in question. This traditional level was developed taking into account the role that the safety-related systems played, in the risk reduction, and the hazardous consequences should such systems fail.

In this generically-based International Standard it is not possible to state what the tolerable risk level should be for a specific application and, consequently what the required Integrity Level should be for the safety-related systems for a specific application. However, it should be possible in application-specific standards to specify exactly what the required Integrity Levels, for the safety-related systems, should be, since all relevant factors should be known. Therefore, in application-specific standards, the required safety integrity level will be specified in the Safety Integrity Requirements Specification and will be in terms of the three system safety elements. For example:-

- the CONFIGURATION will be specified in terms of the number of safety related systems and should indicate how the requirements relating to the programmable electronics (both hardware and software) can be met.

- the SAFETY RELATED HARDWARE RELIABILITY will specify whether a qualitative or quantitative approach should be adopted, and will indicate exactly how this system safety element, in the context of the application in question, can be met.

- the SYSTEMATIC INTEGRITY will specify how far the designer or assessor should go in ensuring that adequate precautions have been built into the design against systematic failure causes.

A major objective in producing this generically-based International Standard was to enable others, to produce their own application-specific Standards. Consequently, when using this International Standard to formulate application-specific standards, or when applying it to an application for which there is no application-specific standard, there will be a need to determine the required level of safety integrity for the overall system.

The concept of conventional safety integrity (see Clause 3.16) is one approach to the development of the Safety Integrity Requirements Specification for the specific application when none exist. In essence, this means that if it is required to use a PES in an application, for which no application-specific standard exists, then much useful information can be obtained by determining the level of safety integrity which has been achieved, in situations similar to those under consideration, by safety-related conventional systems which have traditionally been accepted as good engineering practice. Engineering judgement will be necessary when carrying out this comparison, and if this is borne in mind, the concept can be very valuable.

A.7 Summary

This International Standard separates the concept of risk from that of safety-integrity and uses the terms in the following manner:

- risk applies to the EUC
- safety integrity applies to the total configuration of safety-related systems and to individual safety-related systems
- software integrity applies to the software in individual safety-related systems.

The concept of Integrity Levels are developed. The Integrity Levels specify the level of safety-integrity required for the total configuration of safety-related systems and individual safety-related systems.

The specific system requirements, for the safety-related systems, for each Integrity Level are specified in terms of the system safety elements (see Clause 3.15).

Fig. A1 Risk Reduction: General Concepts

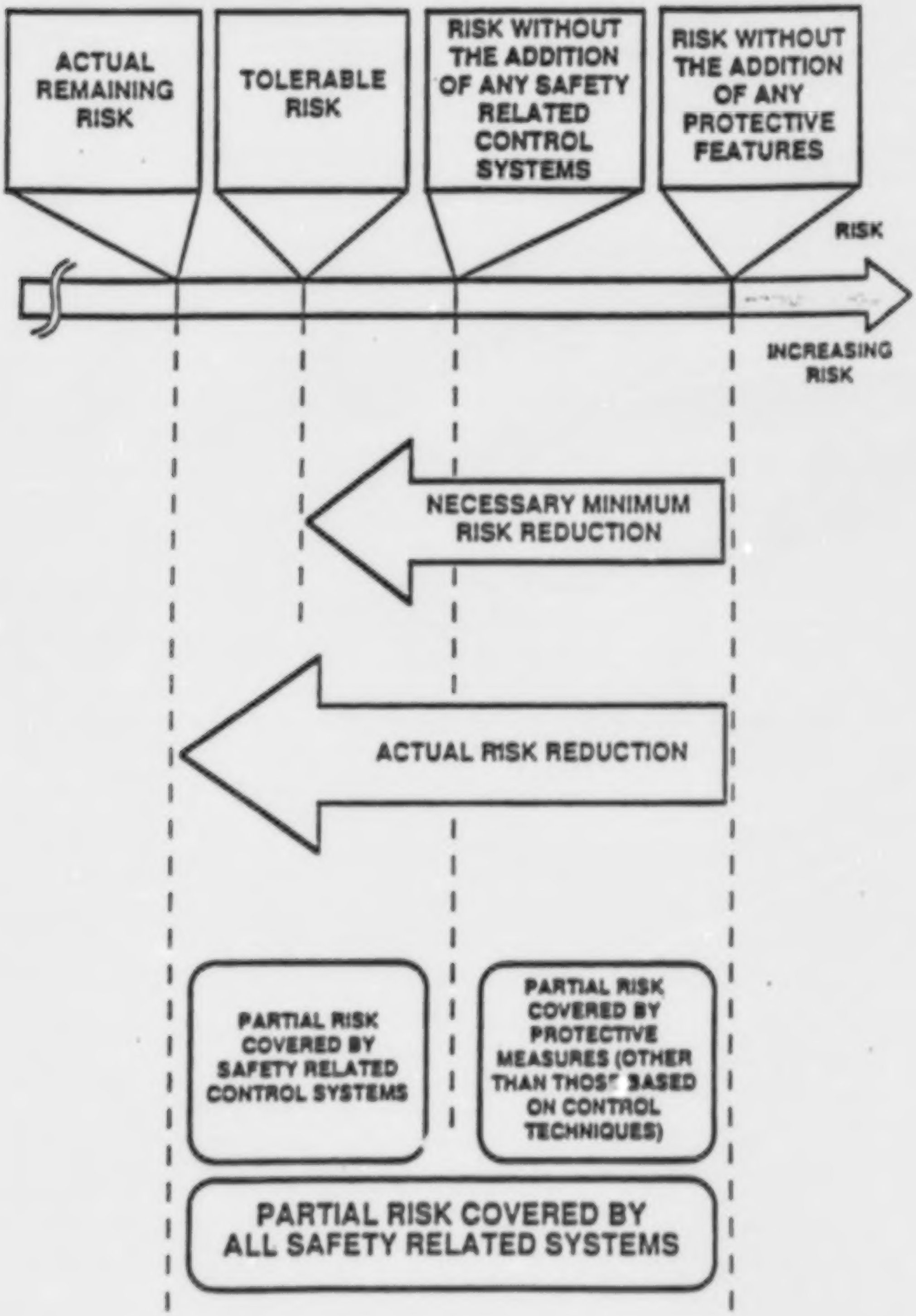
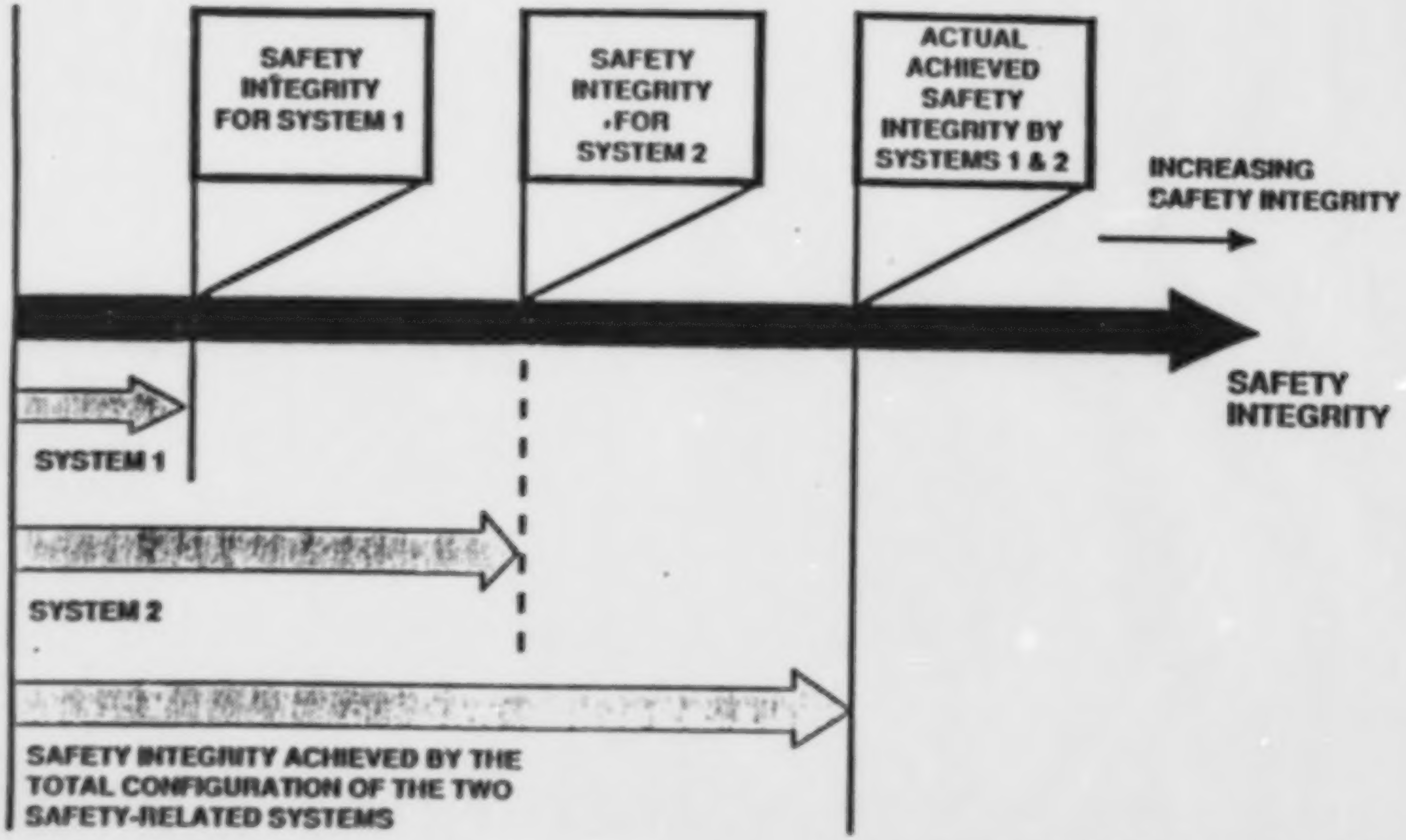

ACTUAL REMAINING RISK
TOLERABLE RISK
RISK WITHOUT THE ADDITION OF ANY SAFETY RELATED CONTROL SYSTEMS
RISK WITHOUT THE ADDITION OF ANY PROTECTIVE FEATURES
RISK
INCREASING RISK
NECESSARY MINIMUM RISK REDUCTION
ACTUAL RISK REDUCTION
PARTIAL RISK COVERED BY SAFETY RELATED CONTROL SYSTEMS
PARTIAL RISK COVERED BY PROTECTIVE MEASURES (OTHER THAN THOSE BASED ON CONTROL TECHNIQUES)
PARTIAL RISK COVERED BY ALL SAFETY RELATED SYSTEMS

Fig A2 Safety integrity concepts: simple example to illustrate safety integrity terminology for a total configuration of two safety-related systems


SAFETY INTEGRITY FOR SYSTEM 1
SAFETY INTEGRITY FOR SYSTEM 2
ACTUAL ACHIEVED SAFETY INTEGRITY BY SYSTEMS 1 & 2
INCREASING SAFETY INTEGRITY
SAFETY INTEGRITY
SYSTEM 1
SYSTEM 2
SAFETY INTEGRITY ACHIEVED BY THE TOTAL CONFIGURATION OF THE TWO SAFETY-RELATED SYSTEMS

(NOT TO SCALE : ILLUSTRATIVE ONLY)

Fig. A3 SEPARATION OF RISK AND SAFETY INTEGRITY

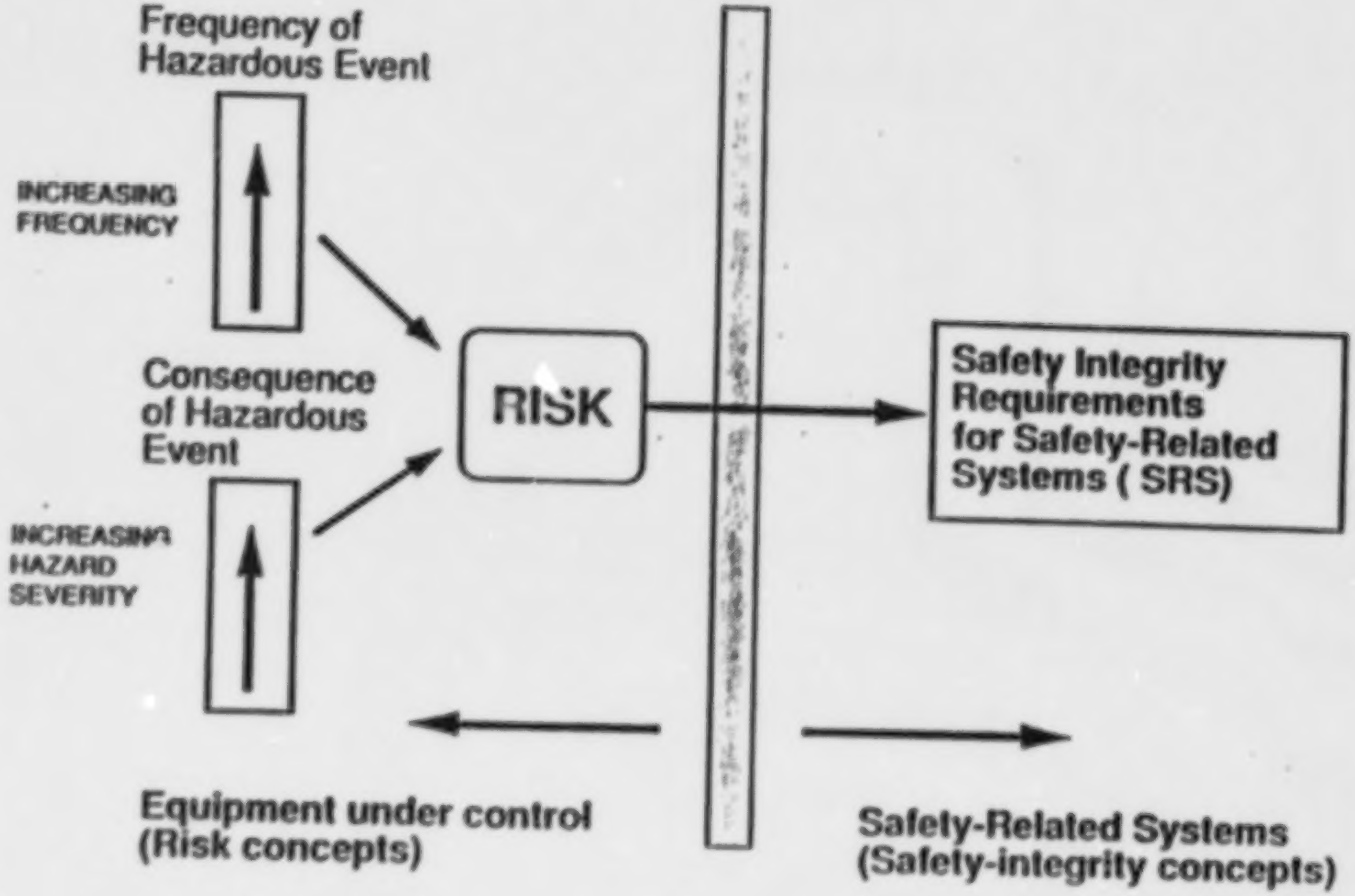
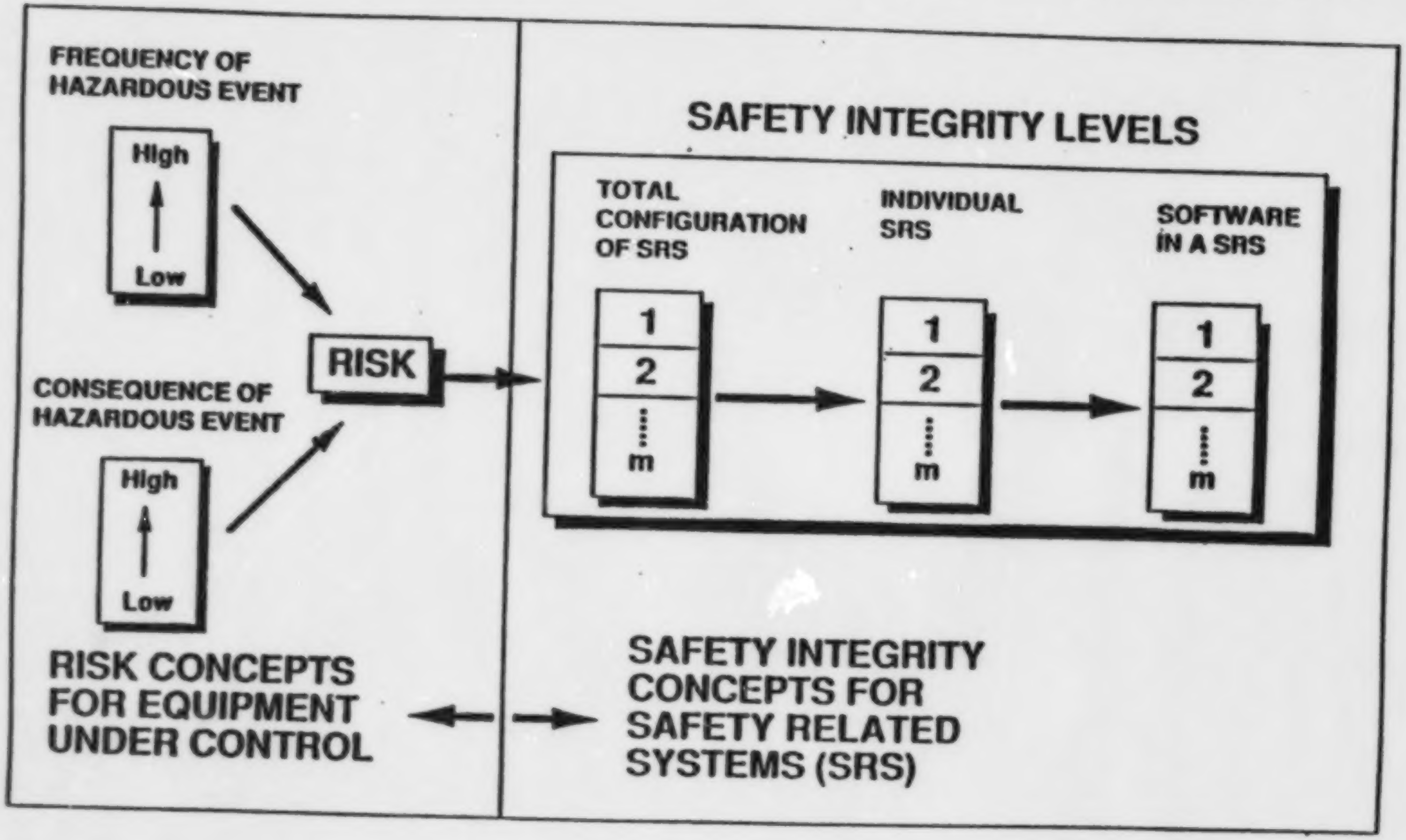

Frequency of Hazardous Event
INCREASING FREQUENCY
Consequence of Hazardous Event
INCREASING HAZARD SEVERITY
RISK
Safety Integrity Requirements for Safety-Related Systems (SRS)
Equipment under control (Risk concepts)
Safety-Related Systems (Safety-integrity concepts)

Fig. A4 RISK AND SAFETY INTEGRITY LEVELS: OVERALL FRAMEWORK


FREQUENCY OF HAZARDOUS EVENT
High
Low
CONSEQUENCE OF HAZARDOUS EVENT
High
Low
RISK
SAFETY INTEGRITY LEVELS
TOTAL CONFIGURATION OF SRS
1
2
m
INDIVIDUAL SRS
1
2
m
SOFTWARE IN A SRS
1
2
m
RISK CONCEPTS FOR EQUIPMENT UNDER CONTROL
SAFETY INTEGRITY CONCEPTS FOR SAFETY RELATED SYSTEMS (SRS)

ANNEX B (Normative)

CONSIDERATIONS UNDERLYING THE GUIDANCE

B1 Introduction

In order to ensure safe operation of safety related PES's, it is necessary to recognise the various possible causes of PES failure and to ensure that adequate precautions are taken against each.

Two categories of failure are considered in this document:

- Random hardware failures, and
- Systematic failures

B2 Random hardware failures

Random hardware failures are those failures which result from a variety of normal degradation mechanisms in the hardware. Since there are many such mechanisms occurring at different rates in different components and since manufacturing tolerances cause components to fail due to these mechanisms after different times in operation, failures of a total equipment comprising many components occur at unpredictable, i.e. random, times.

Measures of system reliability, such as the mean time between failures (MTBF) are valuable in the consideration of system safety although, since programmable electronics are under rapid and continuous development, the reliability/safety data which can be collected on any specific system is often limited.

B3 Systematic failures

However, measures of reliability such as the MTBF are concerned only with random hardware failures and do not include systematic failures. These are due to errors in the design, construction or use of a system which cause it to fail under some particular combinations of inputs or under some particular environmental condition. Any failure of the safety related system, which does not arise from a random hardware failure is, by definition, a systematic failure. Some common examples of causes of systematic failure are given below.

- A PES, or its program, may contain faults caused by errors in specification or design. A PES can rarely be fully tested under all possible conditions and program faults may remain unrevealed until a particular set operating conditions causes a program failure. The failure may cause conditions to be altered dangerously or faulty data to be produced.

- A fault may be introduced into the program or data stored in the memory of the PES, as a result of some transient fault or disturbance.

- Program errors may be introduced by software modifications carried. Since it is usually not possible to exhaustively test the software after a change has been made, such errors may not be detected until a failure results.

- Electrical interference: particular attention needs to be paid to the electromagnetic compatibility.

Systematic failures are particularly important in situations in which more tha one system (redundancy) is provided in order to increase safety. Redundanc systems are effective if failures occur independently in each system since ther will be low probability that more than one system will fail at the same time. Redundancy systems are therefore effective against random hardware failures. However, if each system contains a common fault, it is possible that each syste will suffer a systematic failure under some particular conditions. Thus proportion of systematic failures will be multiple failures. These failure are known as Common Cause Failures (CCF) and they can be a severe limitation t safety achievable by redundancy systems.

Systematic failures can broadly be classified as:

- Systematic failures due to errors or omissions in the system requirement specification;

- Systematic failures due to errors in the design, manufacture, installation operation of the hardware. This would include failures arising fro environmental causes and human error (eg. operator);

- Systematic failures due to faults in the software.

B4 Integrity block diagrams

The concept of systematic failures and random hardware failures can be represented by a number of safety integrity block diagrams (see Figure B.1).

For a non redundant system, the overall safety integrity may be represented by a block diagram as indicated in Fig B.1(a). Failure of this system occurs if one or more of the series blocks fails. It is therefore necessary to ensure that adequate safety integrity be demonstrated for each of the four aspects.

The block diagram for a system with two identical redundant programmable electronics with identical software, operating under identical conditions is shown in Fig B.1(b). In this case, the system fails when both programmable electronics fail, due to random hardware failures (B_1 and B_2), due to systematic failures (C_1 and C_2) or due to a combination of hardware and systematic failures (B_1 and C_2 or B_2 and C_3). However, the system may also fail due to a single specification error (A); a single systematic failure affecting both programmable electronics (CCF); or a single software failure (D).

Note: Failure of both systems due to an incorrect Safety Requirement Specification is classified as a common cause failure (CCF). The CCF box in Figure B.1(b) indicates the common cause failures arising from other systematic failure causes.

Redundancy principles, implemented in the hardware, can be a useful strategy for improving the safety integrity. However, the gain in safety integrity may be limited by the possibility of common cause failure mechanisms affecting some or all of the channels/systems.

Fig B.1(d) is a block diagram for two diverse systems. System 1 is a programmable electronic system; System 2 is a non-programmable system (eg. electromechanical/pneumatic/hydraulic system). The Safety Requirements Specification is a potential course of common cause failure. However, other common cause failures (shown by the "CCF" box are likely to be very low because of the large degree of diversity that has been applied.

The use of diversity as a protection against the effects of software faults depends upon the probability that the different sets of software do not each contain common cause software faults (see Figs B.1(c) and B.2(d); i.e. faults which would cause similar failures of each set of software under the same set of operating conditions. This probability depends upon the extent to which diversity is applied and in practice it is difficult to achieve total diversity.

B5 Summary

The above considerations show that it is necessary, in the context of the overall system safety integrity to consider not only hardware reliability with respect to random hardware failures but also systematic integrity with respect to systematic failures. Consideration can be given in quantified terms to random hardware failures, but the frequency of systematic failures cannot be quantified. Therefore, the safety integrity of a PES cannot be considered solely in quantified terms.

The system safety elements, configuration, safety related hardware reliability and systematic integrity and are intended to provide adequate precautions against both random hardware failures and systematic failures. This is illustrated in Figure B.2.

FIGURE B.1:

SAFETY INTEGRITY BLOCK DIAGRAMS

A = Errors or omissions in the Safety Requirements Specification
B = Random Hardware Failures
C = Systematic (Design, Manufacture, Installation and Operation) Failures
D = Software Failures
CCF = Common Cause Failures
CCF*= Common Cause Failures Software

Figure B.1(a): Block diagram representing 4 failure causes for a non-redundant system.

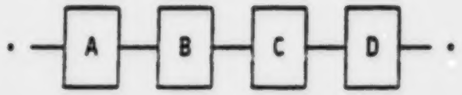


Figure B.1(b): Block diagram for a two systems with dual redundancy -because of the redundancy an additional factor, that of a single systematic failure affecting both programmable electronics (Common Cause Failure, CCF) has to be considered.

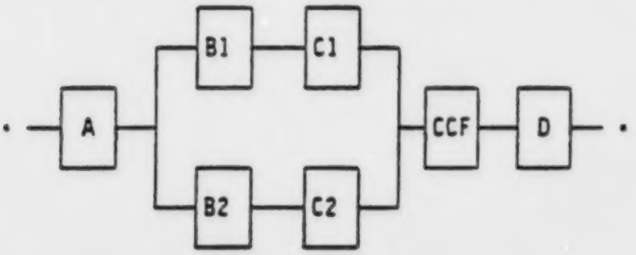


Figure B.1(c): Block diagram of two diverse systems. System 1 is a programmable electronic system; System 2 is a non-programmable system (eg. electromechanical/pneumatic/hydraulic system)

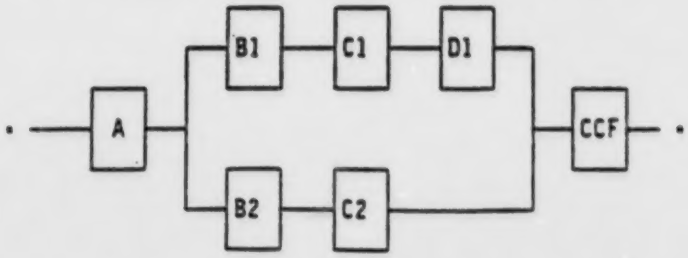


Figure B.1(d): Block diagram representing a two systems of programmable electronics, the hardware of which may be different from each other, with each using different (diverse) software design implementations.

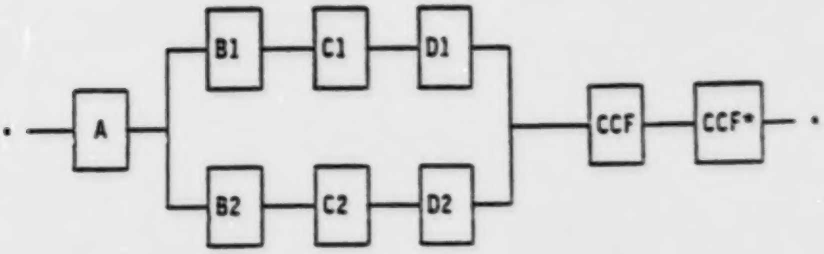
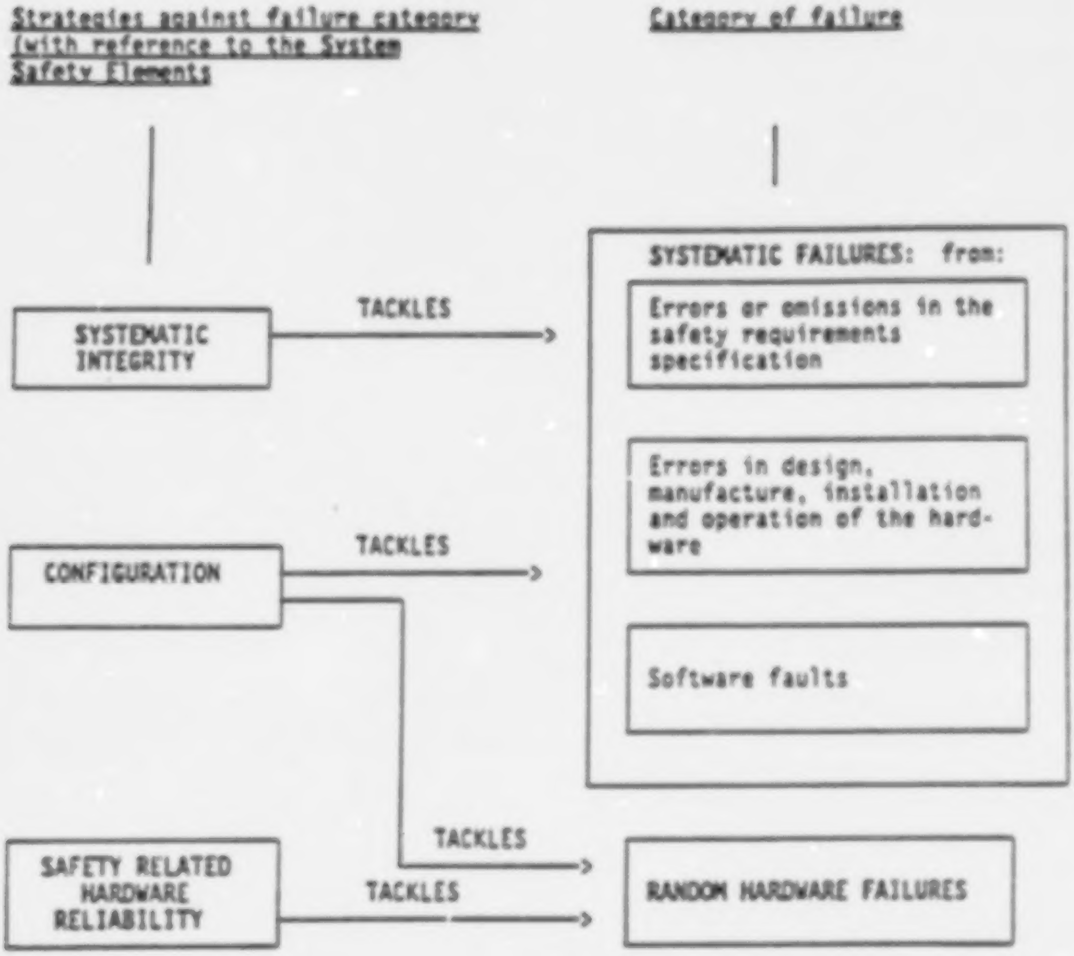


FIGURE B.2: DESIGN AND ASSESSMENT STRATEGY

This diagram illustrates how, by means of the system safety elements, the two categories of failure are addressed.


Strategies against failure category (with reference to the System Safety Elements
Category of failure
SYSTEMATIC FAILURES: from:
SYSTEMATIC INTEGRITY
TACKLES
Errors or omissions in the safety requirements specification
Errors in design, manufacture, installation and operation of the hardware
CONFIGURATION
TACKLES
Software faults
TACKLES
SAFETY RELATED HARDWARE RELIABILITY
TACKLES
RANDOM HARDWARE FAILURES

ANNEX C (informative)

EXAMPLES OF DIFFERENT PESs TO ILLUSTRATE EMBEDDED AND APPLICATIONS SOFTWARE

C.1 Microprocessor based analog (three-term) controller

These are designed as replacements for conventional analog controllers. Algorithms for the calculation of the output level according to the three term controller equations are held in permanent memory (ROM). Usually, the user can change only the three-term controller constants and there is no applications software. The embedded software is, usually, relatively simple and well proven by testing and by wide application.

C.2 Programmable logic controller

A programmable logic controller (PLC) is usually programmed in a symbolic language designed for simple programming from the equivalent relay logic ladder diagrams or from the ladder diagram itself. The program is sometimes developed and compiled on an external unit which may also transfer the machine code to permanent memory for use in the PLC. The PLC itself may have embedded software to load and run the program, and for self test.

C.3 Programmable controller

A programmable controller (PC) can perform multi-loop analog control and/or sequence control. They invariably include embedded software in permanent memory for various purposes including communications with peripherals and other PCs preprogrammed control algorithms, program development and operating system. The amount of application specific software varies considerably. Purely analog controllers may require only the setting of process control parameters but sometimes a complex sequence control program may be required.

C.4 Numerically controlled machines and robots

The applications program is simply a list of positions that each axis or moveable part of the machine must move to, perhaps qualified by speed instructions or 'IF instructions. Embedded software in the machine interprets the instructions and controls the necessary movement with a feedback control algorithm. In the 'teach' mode of robots, the machine moves directly from the operator commands and stores the attained positions as the set of required target positions. The embedded software may be distributed: eg robots sometimes have processors dedicated to the control of each axis.

ANNEX D (informative)

SYSTEM LIFECYCLE MODEL

D.1 Introduction

The control of a system from initial conception to the end of its use requires that there be a description of the system lifecycle of the system giving a list of actions to be taken at each phase of the lifecycle. The lifecycle model divides the life of the system into phases, each terminated by a review - forming a milestone in the life of the system. The system lifecycle model is shown diagramatically in Figure D.1.

A short explanation of what is meant by each phase is indicated in Clauses D.2 to D.8.

D.2 Study phase

This phase begins with the user's specification of requirements. It culminates with a specification, agreed between the principal (customer) and the supplier. This specification describes the functions, facilities and interfaces to be provided by the system and the required performance and safety integrity level.

D.3 Design phase

In this phase the functional specifications are translated into specifications for the various hardware and software system parts. Subsequently, all techniques, procedures and standards by which these specifications will be implemented are exactly defined. Selection of components are part of this task. Some components may have a high level of complexity and may be regarded as complete system parts (referred to as "bought-out equipment" in Figure D.1). On selection of such parts, care must be taken to ensure that their application is evaluated against (safety and reliability) requirements of the total system.

D.4 Implementation phase

This phase represents the execution of the implementation plan which is defined in the design phase. The phase ends with a check on conformance. At the end o this phase all system elements are individually present and conform to the specification.

D.5 Integration phase

This phase is meant to obtain, prior to shipment, confidence in the system as whole. The system elements are built together and the system is tested against the functional specifications in a simulated environment usually at the place manufacturing.

D.6 Installation phase

This phase starts with installation of the system in the real environment. The phase ends when sufficient confidence is gained that the system will fulfill all user requirements and all specification items. In order to achieve this, specification items are tested in the real environment and operational procedures are tested in parallel runs.

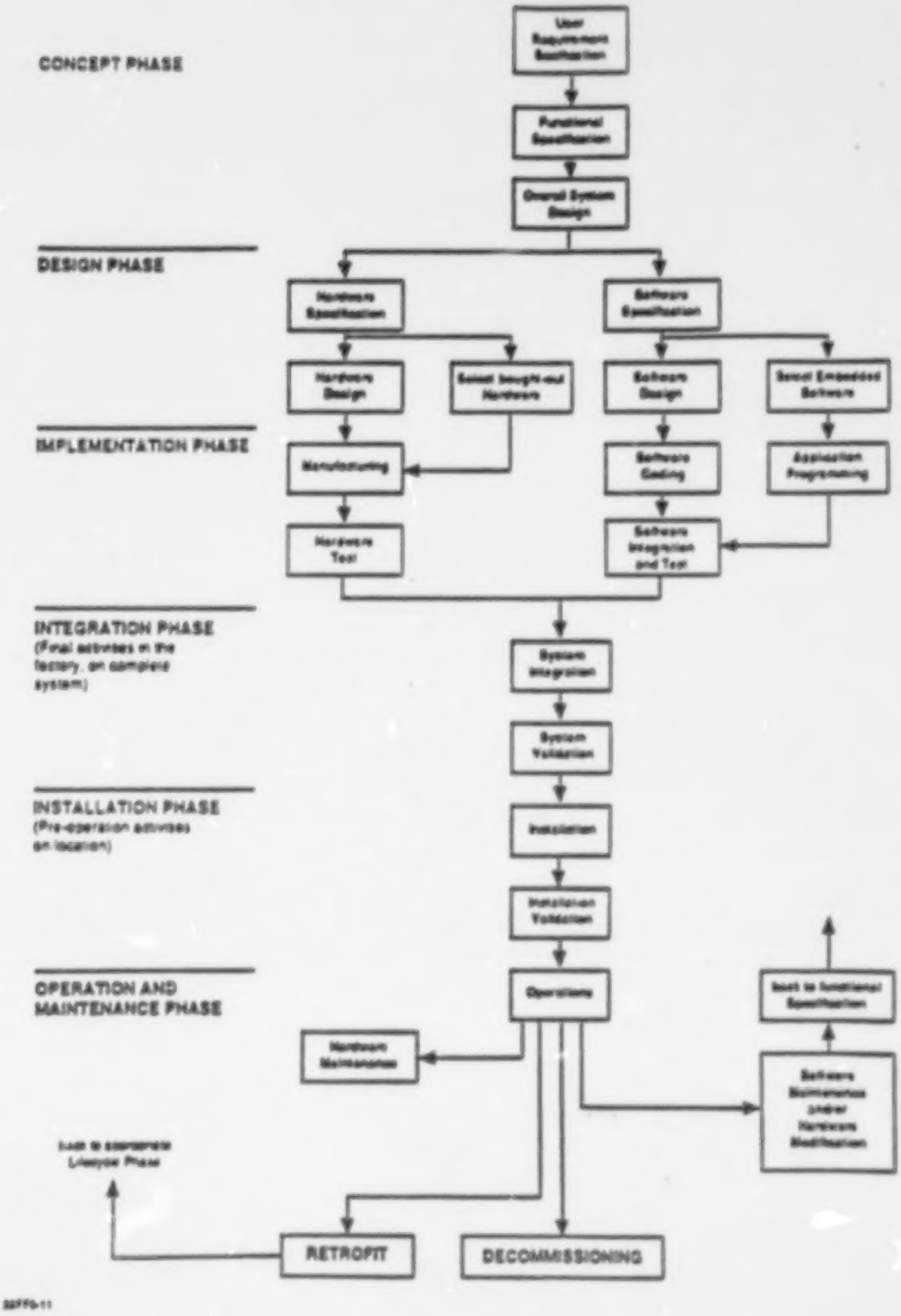
D.7 Operation and maintenance phase

During operation the need will exist to maintain the system. Because of possible random hardware degradation mechanisms preventive and corrective actions are carried out according to the hardware maintenance procedures. Systematic failures and shortcomings due to errors in design, construction or use of the system will require modifications to be made in the system design - as will additional requirements. In order to implement such modifications the steps of the life cycle, starting from the functional specification, must be recognised and addressed.

D.8 Decommissioning

This phase relates to the control of the gradual reduction of facilities, either for decommissioning of the EUC, or for a total replacement of the PES. The phase concludes with a total shutdown of the PES.

Fig. D1 System Lifecycle Model


User Requirement Specification
Functional Specification
Overall System Design
CONCEPT PHASE
DESIGN PHASE
Hardware Specification
Software Specification
Hardware Design
Select bought-out Hardware
Software Design
Select Embedded Software
IMPLEMENTATION PHASE
Manufacturing
Software Coding
Application Programming
Hardware Test
Software Integration and Test
INTEGRATION PHASE
(Final activities in the factory, on complete system)
System Integration
System Validation
INSTALLATION PHASE
(Pre-operation activities on location)
Installation
Installation Validation
OPERATION AND MAINTENANCE PHASE
Operations
Back to functional Specification
Hardware Maintenance
Software Maintenance under Hardware Modification
Back to appropriate Lifecycle Phase
RETROFIT
DECOMMISSIONING
32770-11

ANNEX E (Informative)

CASE STUDY TO ILLUSTRATE ASPECTS OF THE SAFETY LIFECYCLE MODEL

E.1 General description of equipment and systems

Figure E.1 shows a control and protection scheme for a process vessel. There is a critical point beyond which the liquid level must not rise if a serious hazard is to be avoided. The EUC has a large number of safe states and the purpose of the protection scheme is to enable one of these states to be reached by stopping liquid flow. A general description of the EUC, and associated control and protection scheme, is as follows.

The process vessel has, on its lower part, a level transmitter feeding into the process computer. The signal from the level transmitter provides:-

- Control of the valves which control the liquid in the vessel.
- Indication, for the process operators, of the level of liquid in the vessel.
- An alarm set at a prescribed value, eg. 80% level. This alarm is routed via the process computer.

Above the level transmitter is a single level switch which is hard-wired into an alarm.

Above the single level switch are three level switches feeding into two programmable controllers (PC1 and PC2). Within each PC "2 out of 3" voting takes place. The output of each PC goes to the trip valves and also to a trip alarm.

Note 1: The two programmable controllers are used for carrying out many safety functions - only one of which is dealt with in this particular example. Such an arrangements would not usually be used for solely carrying out the safety functions associated with this specific hazard.

Note: 2) The architecture for the redundancy programmable controllers is give by way of example only. Triple redundant controllers with majority voting could *also have been selected. In fact PES architectures for safety related systems will vary widely, depending on the system requirements and the available PES equipment.*

The trip valves can also be operated by direct means from a stop button which is hard-wired.

E.2 Hazard analysis and risk assessment

The HAZARD ANALYSIS, has identified the hazard (filling of the vessel beyond the critical point). A PRELIMINARY RISK ASSESSMENT has determined that the risk level without protective systems, is too high. That is, the combination of the frequency with which the vessel could rupture and the consequences (that is the risk) is above what is deemed to be a tolerable risk level. It is therefore necessary to consider the control and protection scheme that has been developed for the vessel to determine whether a tolerable risk level is achieved.

E.3 Safety related system identification and designation

The process of identifying and designating the safety-related systems is described as follows (see Figure E2).

In the context of the liquid level, from a control and protection viewpoint, there are essentially 4 systems. As the level of liquid in the vessel rises (eg. 80% level) Alarm 1 is raised. The plant has been so designed that the operator can, in receipt of Alarm 1, take corrective action to prevent the liquid level from rising any further and, in fact, to reduce the level. This is system 1.

If, however, the operator cannot control the liquid level, or, the proces computer is incapable of taking the corrective action, even though the operato has performed his functions corractly, the liquid level may rise until the hard-wired Alarm 2 is raised. Even at this point, the operator should still be abl to take corrective action to stop the liquid level rising any further. This i system 2.

If the operator cannot control the liquid level and the liquid continues rising it will then activate, via the 3 level switches and PC1 and PC2) the trip valve to automatically bring the plant to a safe state. This is system 3. The 'Tri

Indication' is also raised to indicate to the operator that system 3 has operated.

Should system 3 fail, then the operator can bring the plant to a safe state by operating a hard-wired shut-down system. The operator obtains information on which to make a decision based on information from systems 1, 2 and 3 but also on Alarm 3, Alarm 4, the ultimate level alarm, forms part of a independent manually operated shut-down system (ie. sensor, alarm, operator, shut-down button hardwired to the trip valves). This is system 4.

Note: *System 4 is shown as having a single sensor. This is show for illustrative purposes only. For the level of safety integrity required for the total configuration of safety-related systems, and in order to avoid spurious shut-downs, it is likely that system 4 would have a higher level of safety-integrity than shown and techniques would be applied to increase its availability by, for example, majority voting techniques using several sensors.*

Although systems 1 and 2 contribute to safety they have not been designated as safety related systems. The reasons mainly centre around the difficulty of assessing the process control computer for safety. Essentially, systems 3 and 4 have to provide the requisite level of safety integrity taking into account factors such as the number of times systems 3 and 4 will be required to operate (that is, the demand rate on the safety related systems) caused by failure of systems 1 and 2. A further critical requirement in the design of systems 3 and 4 is the need to ensure that all reasonably foreseeable failure modes of systems 1 and 2 are identified in order that systems 3 and 4 can each independently put the EUC into a safe state in the event of failure of systems 1 and 2. It is also a necessary requirement that systems 3 and 4 cater for reasonably foreseeable failures of the EUC. These design requirements need to be specified in the Functional Safety Requirements Specification. This specification specifies exactly the functions that have to be carried out by systems 3 and 4. The level of certainty that the functions of systems 3 and 4 should be carried out, to achieve the level of safety required,is contained in the Safety Integrity Requirements Specification.

Therefore in this case:

- Systems 1, 2, 3 and 4 constitute together the total configuration of systems.
- Systems 3 and 4 constitute together the total configuration of safety-related systems
- Systems 3 and 4 are individual, and each fully independent, safety-related systems

It can be seen that whether a system is designated safety-related is determined by the particular circumstances. The role of the operator in systems 1 and 2 is very important, since efficient and correct operator action will minimise the demands on the 2 protection systems (systems 3 and 4). This has an important bearing on the level of safety integrity required of systems 3 and 4 in order to achieve an acceptable hazard rate (ie. the rate at which the hazardous event occurs - which is one element of the risk).

Fig. E1 CONTROL AND PROTECTION SCHEME

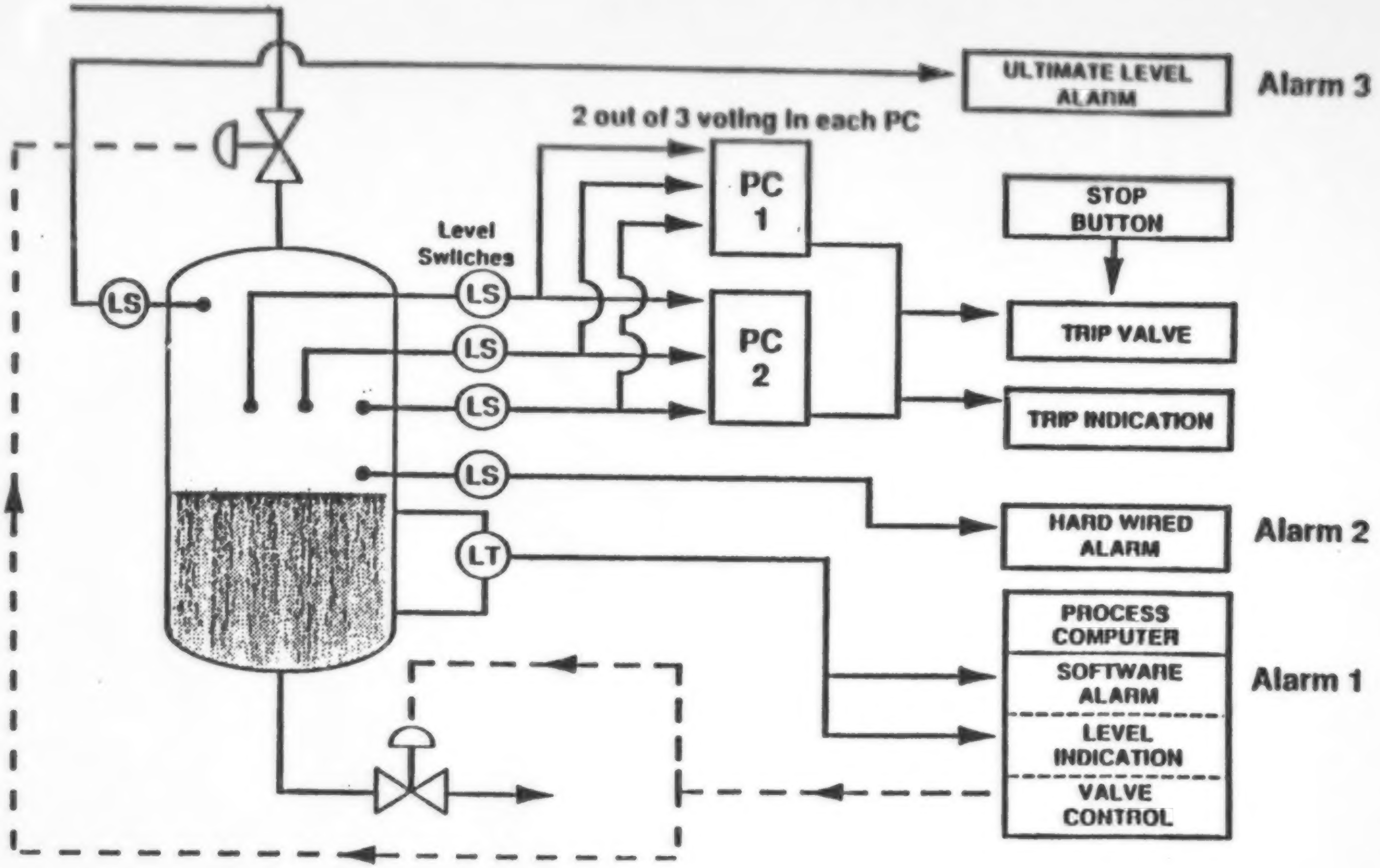
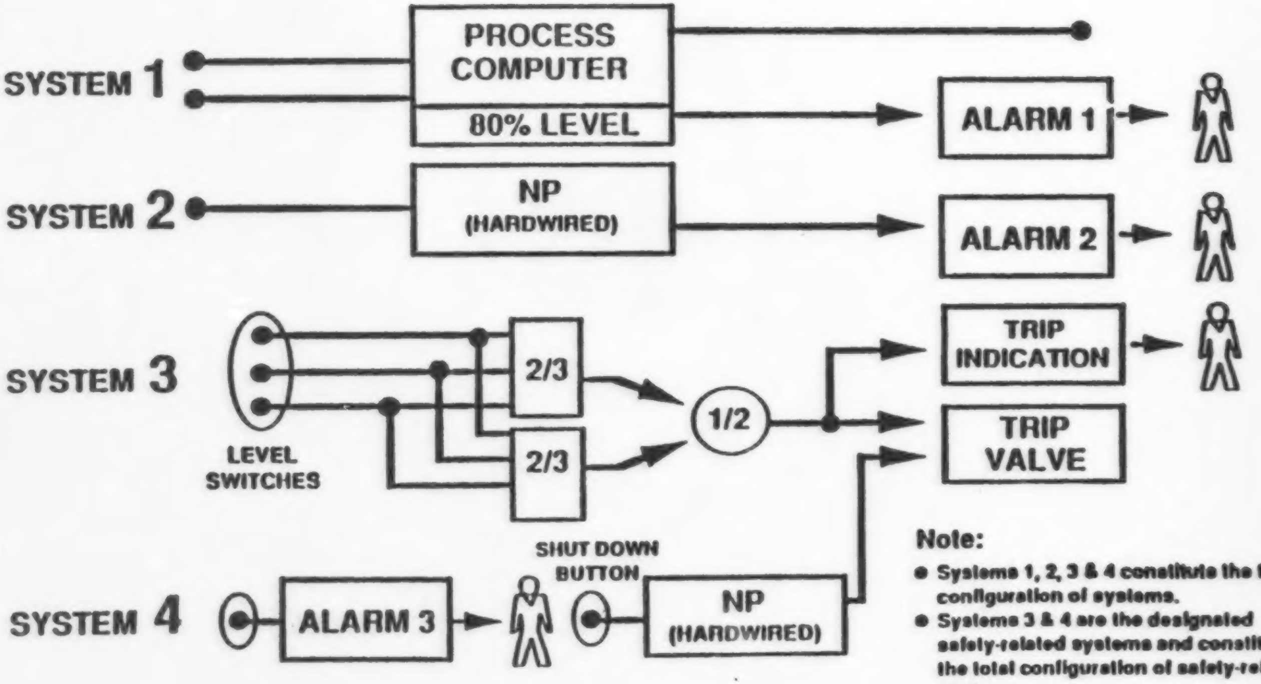

ULTIMATE LEVEL ALARM
Alarm 3
2 out of 3 voting in each PC
PC 1
PC 2
STOP BUTTON
TRIP VALVE
TRIP INDICATION
Level Switches
LS
LS
LS
LS
LS
LT
HARD WIRED ALARM
Alarm 2
PROCESS COMPUTER
SOFTWARE ALARM
LEVEL INDICATION
VALVE CONTROL
Alarm 1

Fig. E2 CONFIGURATION OF CONTROL AND PROTECTION SYSTEMS


SYSTEM 1
PROCESS COMPUTER
80% LEVEL
ALARM 1
SYSTEM 2
NP (HARDWIRED)
ALARM 2
SYSTEM 3
LEVEL SWITCHES
2/3
2/3
1/2
TRIP INDICATION
TRIP VALVE
SYSTEM 4
ALARM 3
SHUT DOWN BUTTON
NP (HARDWIRED)

Note:

- Systems 1, 2, 3 & 4 constitute the total configuration of systems.
- Systems 3 & 4 are the designated safety-related systems and constitute the total configuration of safety-related systems.

ANNEX F (Informative)

RESERVED FOR FUTURE CASE STUDY.

ANNEX G (Informative)

GENERAL DOCUMENTATION REQUIREMENTS

Clause	Contents
G.1	General
G.2	Contract
G.3	Requirements Specification
G.4	Quality Plan - Project description
G.5	Functional Descriptions
G.6	User's guide
G.7	Acceptance Test Plan
G.8	Acceptance Test Description
G.9	Design Description
G.10	Integration Test Description
G.11	Test Description, module
G.12	Code List
G.13	Test Record
G.14	Integration Test Record
G.15	Acceptance Test Record
G.16	Review Meeting Record
G.17	Project Report

6.1 General

All phases of a development project are normally associated with one or several documents. These documents may belong to one of these categories:

- project administration documents
- product documents
- quality documents
- production documents

Project administration documents include quality plans (Project Description, Time Schedule, Resource Plan, Cost Plan), Project Reports.

Product documents include Functional Description, Design Description, Test Description etc.

Quality documents include records from review meetings, Test Records etc.

The production documents are the specific documents necessary for the preparation of, the manufacturing and installation of the product. This document category is addressed in this Annex.

The product documents produced in one phase will normally be input to the next or following phases or to sub-projects.

The intention should be to make the documents as complete as possible in the phase where they are mainly produced. This will increase the possibility of revealing faults as early as possible which have been introduced in the design. However some of the documents will have to be revised during later phases when more details are available. Such revision should be made in an orderly manner and the revisions of the documents should be reviewed by experts and approved in the same way as when the document was first produced.

The above mentioned documents will fulfil the requirement in the standard ISO 9001 for quality systems and should be maintained during the total life cycle of the product. In addition there can be other documents like Users Manuals etc, using the above mentioned documents as sources but edited to suit the different users.

The following Clauses are mainly structured according to the system lifecycle model used in this standard (Annex D) and contain descriptions of the content of the main documents having a bearing on safety. The necessary documents structured in accordance with the safety lifecycle are of the same type, but cover the safety aspects of the system. They can be separate documents or separate chapters in the normal system documents. The descriptions are generic and considered to act as a general definition of the different document types. It has to be detailed for the different types of projects and products. Not all documents are necessary in all cases and it may be practical to integrate the information described for several documents into one. For some documents where the character of the content is totally different for systems and for modules the description must be split up.

The documents mentioned here, their content and names are not intended to be a part of this International Standard. The primary objective of this Annex is to complement this Standard. The document names and the way the content is split up in the different documents can vary in different standards and specific organizations.

6.2 Contract

The Contract is a document with the agreements between the supplier and the customer. It should directly or by references to other documents specify function, cost, time schedule, warranties customer and suppliers representatives. The contract has to be approved by both parties.

6.3 Requirement Specification - Safety requirements specification

The safety requirements specification can be a separate document or a separate chapter in the system requirement specification. This part describes the safety requirements.

The Requirement Specification is the input to the management for the decision to start the project and to the project team giving the framework for the project. It can be practical to split up the Requirements Specification according to the different system levels:

- Total PES
- Hardware system
  - system
  - modules
- Software system
  - standard computer software (operating- system etc)
  - application software

The Requirements Specification defines the technical requirements and the function of the product to be developed. It should describe the function and not the design (realisation of the function). The following is an example of items to be described:

- Function
- Data structures
- Different versions achieved by parameter settings etc
- Physical interfaces and functional interface requirements

The Requirement Specification is the input to the project where it will be further detailed during the definition phase and used to create a Functional Description. When the Functional Description has been verified against the Requirements Specification, the latter has usually fulfilled its role. Any further revision relating to the function is dealt with by revising and reviewing the Functional Description.

The Requirements Specification can be produced by the customer or by the customer in co-operation with the supplier. For standard systems that could be delivered to different customers it may be produced entirely by the supplier.

6.4 Quality Plan: Project Description

The Quality Plan is a set of project documents describing the project: the premises, the organization, project plans, objectives, tools, rules etc to be used within the project. It can contain references to standard tools and rules used in the organization for that type of project. The following documents shall be produced for each project:

- Project Description
- Time Schedule (not described here)
- Resource Plan (not described here)
- Cost Plan (not described here)

The Project Description is the top project document containing the following information:

- General
    - background
    - goal (including confirmation of cost goal for the final product)
    - organizational relations (line organization, other projects)
    - handling of Project Description (approval, updating etc)
- Scope of function (reference to the Requirement Specification)
- Registration and distribution of documents
- Project organization
    - project members
    - steering group
    - review groups
    - customer representatives
- Project planning (references to time schedule, resource and cos plans)
- Development prerequisites/methods (methods and tools to be used special difficulties
- Termination of the project (criteria for termination and how)

6.5 Functional Description

The Functional Description is normally split into a hierarchy according t the different system levels in the same way as the Requiremen Specification:

- Total PES
- Hardware
  - system
  - modules
- Software
  - standard computer software (operating system etc)
  - application software

The top document on the PES level describes the overall function of the total PES, identify the different parts (hardware and software) and describes the overall function including the application, the process and control function.

The Functional Specification for the hardware describes the computer hardware, the split up into different units and the transmitters and actuators. The document will normally be split up into a description on the system level and descriptions of the different units.

The Functional Description for software will normally be split up into standard system software (e.g. operating system) and application software. It will describe the total function down to the lowest level, or the description can be split up into descriptions of sub systems and modules giving the details. The Functional Description can in the first case be used as the Requirement Specification, function, for the lower levels.

The Functional Description will be input to the activities to produce:

- Design Description,
- Acceptance test Plan
- Acceptance test Description
- User's guide

6.6 User's Guide

The User's Guide is a product document describing for the user how to use and/or operate the product. The User's Guide can be a supplement to the Functional Description. It is in some cases easier to explain the function by explaining how to use it. In that case the User's Guide has to be reviewed and approved together with the Functional Description.

The User's Guide can be produced in several steps and appear in different shapes. The first produced by the design team is a link in the quality assurance function. It can be re-edited and get different-layout. The correctness has to be assured if the document is re-edited.

6.7 Acceptance Test Plan

The Acceptance Test Plan is a product document describing the structure of the acceptance test.

An acceptance test can be made in steps. Each individual module is tested. After that the subsystem test is focused on the interconnections and cooperation between the individual modules and not on the functions tested on module level. Combination of these methods can be the most optimal. After that the subsystems are integrated into systems and tested in similar ways.

The final acceptance test of the PES can be split up into factory test and site test after installation.

The Acceptance Test Plan should describe this structure, split up, of the acceptance test.

For hardware it may be practical to split up the different parts of acceptance test:

- Functional test
- Climatic tests
- Mechanical tests
- Electric interference test.

This split up should be described in a Acceptance Test Plan.

The Functional Description (and User's Guide) will be the input to producing the Test Plan.

The Acceptance Test Plan will be input to the activities to produce Acceptance Test Descriptions.

6.8 Acceptance Test Description

The Acceptance Test Description is a product document describing the test activities and contains the following:

- General (introduction and identification of the function)
- References (to other documents such as the Functional Description)
- Scope of the test
- General tools and test equipment
- References to documents (Acceptance Test Plan, Functional Description etc)
- Personnel responsible for test and approval
- Test equipment specification (or references to other documentation)
- Rules for Test Record disposition
- The test procedure (step by step).

The acceptance test can occur on different levels, on part of the system, factory acceptance test and site acceptance test.

6.9 Design Description

The Design Description is a product document describing the design of the function described in the Functional Description. The structure and content of the Design Description will differ depending on the system level concerned. The Design Description is used as input to the design and maintenance of the system or unit.

The Design Description should describe the realisation of the function described in the Functional Description. The Functional Description can in some cases be sufficient input to the next phases and sub projects and for the maintenance. Together with the description of sub-systems and units it can be a complete description of the design.

The Design Description is normally split into a hierarchy according to the different system levels in a similar way as the Functional Description. The hierarchy may not be the same as in the Functional Description, but the functions should be traceable in the design:

- Total PES
- Hardware
    - system
    - modules
- Software
    - standard computer software (operating system etc)
    - application software

The top document on PES level will describe the overall design of the total PES (hardware and software) and describe the overall design including the application, the process and control function.

The Design Description for the hardware will describe the configuration of the computer hardware and the transmitters and actuators. The document will normally be divided into a description on system level and descriptions of the different units. Drawings such as circuit diagrams, layout drawings, assembly drawings etc will act as Design Description.

The Design Description for software will normally be divided into standard system software, e.g. operating system, and application software.

For software modules the Design Description may include plain text or flowcharts, pseudo code and other types of textual and graphics tools for design description. It should contain a description of the structure of the software and be an input to the coding and to the maintenance of the module.

The Design Description will be the input into the activities for producing the system.

6.10 Integration Test Description

The Integration Test Description is a product document describing the step by step integration and testing of the different modules and sub-systems into a complete sub-system or system. This is done by the development team to ensure that the different parts will function as a system. The tests can require special tools, program and equipment to simulate the final environment on each step.

The input to produce the Integration Test Description will be the Functional Description and the Design Description.

The Integration Test Description should contain the following information:

- General (introduction and identification of the function)
- References (to other documents such as the Functional Description)
- Overview of the integration steps
- General tools and test equipment
- Rules for integration Test Record disposition
- Integration step n (Detailed description of the integration step)
    - scope of the integration step
    - special tools and equipment
    - activity
    - expected result

Depending on the implementation of the application software and the project organization, the integration test may be split up into integration of the standard system an the application system.

G.11 Test Description: Module

The Test Description is a product document describing the test of the unit, hardware or software, produced by the developer to verify the design before it is integrated with other parts. The tests may require special tools, programs and equipment to simulate the final environment.

The inputs to produce the Test Description are the Functional Description, and the Design Description.

The Test Description should contain the following information:

- General (introduction and identification of the function)
- References (to other documents like the Functional Description)
- Overview of the test
- General tools and test equipment
- Test Record disposition
- Test step n (detailed description of the test step)
  - scope of the test step
  - special tools and equipment required
  - activities
  - expected results

G.12 Code List

The Code List is a product document containing the source- code for the production of the machine code to be used in the target system. The inputs to the production of the code are the Functional Description and the Design Description.

The Code List shall have a heading containing unambiguous identification, inclusive version, of the code and the function it is implementing. It shall use the conventions of the programming language selected, be well structured and contain necessary comments.

G.13 Test Record

The Test Record is a quality document with the results from the test of the unit, code module or hardware unit, during the development. It shall contain the following:

- Unambiguous identification of the module tested
- Person(s) carrying out the test
- When the test was done
- References to Test Description (including version)
- Results from the test
- Resolution of any text failures

6.14 Integration Test Record

The Integration-Test Record is a quality document with the results from the integration of the system during the development. It should contain the following:

- Unambiguous identification of the system (included version)
- Person(s) who carried out the test
- When the integration was done
- References to Integration Test Description (including version)
- Results from the integration
- Resolution of any text failures

6.15 Acceptance Test Record

The Acceptance Test Record is a quality document with the results from the acceptance test of the system. It shall contain the following:

- Unambiguous identification of the system (included version)
- Person(s) who carried out the acceptance test
- When the acceptance test was done
- References to Acceptance Test Description (including version)
- Results from the acceptance test.
- Approval of the Acceptance Test Record by authorized person.

6.16 Review Meeting Record

The Review Meeting Record is a quality document with the results from the review meetings on documents, code lists etc.

The document shall have the following information:

- Unambiguous identification of the item reviewed (including version)
- The review meeting members
- Date of the meeting
- Remarks on the item reviewed.

6.17 Project Report

The Project report is a project document produced by the project leader on a periodic base. The report shall contain the following information:

- General introduction
- Time schedule
- Resource schedule
- Cost schedule
- Special difficulties
- Decisions to be made

The report shall describe the status of the different schedules and comment on differences in planned and actual values.

The Final project report is a special report made when the project is finished in a normal termination or after a decided termination at an earlier stage of the project.

This project shall summarize the results and clearly identify the organisation responsible for the take over of the product developed.

ANNEX H (Normative)

ABBREVIATIONS USED IN THIS INTERNATIONAL STANDARD

ABBREVIATION	FULL EXPRESSION	DEFINITION AND EXPLANATION OF TERM
EUC	Equipment Under Control	See Clause 3.1
PES	Programmable Electronic System	See Clause 3.3
PE	Programmable Electronics	See Clause 3.4
SRS	Safety-Related System	See Clause 3.5
NP	Non-Programmable	See Clause 3.14
CCF	Common cause failure	Annex B
PC	Programmable Controller	

ANNEX I (Informative)

BIBLIOGRAPHY [UNDER CONSIDERATION]

Reference:

1. Fault Tree Analysis - Concepts and Techniques; Fussel J; in Generic Techniques in Reliability Assessment; Henley E and Lyne E (editors); Noordhoff Publishing Co. Leyden, Holland; 1976.

2. MIL-STD-1629A; Procedures for performing failure mode and effect analysis.

3. A Guide to Hazard and Operability Studies; Chemical Industry Safety and Health Council of the Chemical Industries Association, Alembic House, 93 Albert Embankment, London SE1 7TU, UK.

4. Dependability of critical computer systems 1: Edited by F J Redmill;Elsevier Applied Science Publishers. (Guidelines produced by the European Workshop on Industrial Computer Systems; Technical Committee 7 (EWICS TC 7).

5. Programmable electronic systems in safety-related applications: General technical guidelines; ISBN 0 11 88390 63. Published by the Health and Safety Executive, UK. Available from HMSO Publications Centre, PO Box 276, SW8 5DT, UK.

6. DIN V 19250; Grundlegende Sicherheitsbetrachtungen fur MRS - Schutzeinrichtungen; January 1989.

7. British Standard BS 5304: 1988; Safety of Machinery.

8. Document CEN/TC 114/JWG 6-CLC/TC 44X; N48 August 1989: "Categories of Control".

9. DIN V VDE 0801; Principles for computers in safety-related systems.

10. Safety classification of controls; comments on the German approach with a view to European Standardisation. Measurement and Control, Vol 21, November 1988; pages 277-280.

Reproduced in Switzerland
Central Office of the IEC
3 rue de Varembe
GENEVA

[illegible](Secretariat)96

APPENDIX D

Draft British Standard on
Software for Computers in the Application of
Industrial Safety-Related Systems

MR BELL

BSI

Document 89/33006 DC

DRAFT FOR PUBLIC COMMENT

Latest date for receipt of comments

31st January 1990.

British Standards Institution

Our ref PCL/1/1

Date 6th DECEMBER 1989.

Sub-committee PCL/1/1
System considerations
(of Technical Committee PCL/1
Measurement and control)

DRAFT BRITISH STANDARD 0000 : SOFTWARE FOR COMPUTERS IN THE APPLICATION OF INDUSTRIAL SAFETY-RELATED SYSTEMS

65A(SECRETARIAT)94

6066

18 DEC 1989

This draft is now available for public comment and your views and technical comments on it would be appreciated. If you have no specific comments to make but find it generally acceptable it would be helpful if you would notify us accordingly. Suggestions entailing revision of the text should indicate the preferred wording. Please quote the relevant clause number against any comment.

The co-ordination of the requirements of this draft with those of any related standards is of particular importance and you are invited to point out any areas where this may be necessary.

All comments should be sent to the Committee Secretary MR J E TAYLOR at the London/~~Manchester~~ address below. The comments received will be passed to the committee concerned for their consideration; no acknowledgement will normally be sent.

Requests for further copies of this draft should be sent to:
Sales Administration (Drafts), BSI, Linford Wood, Milton Keynes MK14 6LE.
Price £5.00 to Subscribing Members, who will be invoiced.
Non-members should send a remittance of £[illegible] together with an addressed label. Orders not accompanied by a remittance will be subject to a delay.

THIS IS A DRAFT AND MUST NOT BE REGARDED OR USED AS A BRITISH STANDARD

Head Office
2 Park Street London W1A 2BS Telephone 01-629 9000
Telex 266933 BSILON G Head Office
825777 BSIMK G Sales & Accounts

Manchester Office
3 York Street Manchester M2 2AT
Telephone 061-832 3731
Telex 668969 BSIMAN G

Form 38 Part 1 – [illegible]

BSI COMMITTEES TO WHOSE MEMBERS THIS DRAFT HAS ALSO BEEN SENT

STANDARDS COMMITTEE (CHAIRMAN AND SECRETARY ONLY)

PCL/-	INDUSTRIAL-PROCESS MEASUREMENT AND CONTROL STANDARDS POLICY COMMITTEE

TECHNICAL COMMITTEE PCL/1 MEASUREMENT AND CONTROL
TECHNICAL COMMITTEE PEL/84 ELECTRICAL EQUIPMENT OF INDUSTRIAL MACHINES

SUB-COMMITTEE PCL/1/1

SYSTEM CONSIDERATIONS

CONSTITUTION

ORGANIZATIONS REPRESENTED

ELECTRICITY SUPPLY INDUSTRY IN ENGLAND AND WALES
ENERGY INDUSTRIES COUNCIL
ENGINEERING EQUIPMENT AND MATERIALS USERS ASSOCIATION
GAMBICA (BEAMA LTD)
HEALTH AND SAFETY EXECUTIVE
INSTITUTE OF MEASUREMENT AND CONTROL
SIRA LIMITED

BSI
STANDARDS

2 Park Street
London W1A 2BS
Telephone: 01-629 9000
Telex: 266933 BSILON G
Fax Gr 2/3: 01-629 0506

Document: 89/33006 DC

Our ref PCL/1/1

Date November 1989

DRAFT FOR PUBLIC COMMENT

Sub-committee PCL/1/1
System considerations
(of Technical Committee PCL/1
Measurement and control)

DRAFT BRITISH STANDARD 0000 : SOFTWARE FOR COMPUTERS IN THE APPLICATION OF INDUSTRIAL SAFETY-RELATED SYSTEMS

NATIONAL FOREWORD

This draft British Standard has been prepared under the direction of the Industrial Process Measurement and Control Standards Committee.

It is identical with document:

65A(Secretariat)94 : Draft. Software for computers in the application of industrial safety-related systems.

TERMINOLOGY AND CONVENTIONS

Because the text of a draft International Standard has been copied some terminology and certain conventions are not identical with those used in British Standards.

Compliance with a British Standard does not of itself confer immunity from legal obligations.

NOTE. This draft Standard is identical with a document issued by the International Electrotechnical Commission and is based on International discussion in which the UK has taken an active part. Technical comments on this draft will assist in the preparation of the International Standard. Subject to committee approval it is expected that the completed International Standard will be implemented in the UK by the publication of an identical (dual-numbered) British Standard. The draft National Foreword reflects this expectation, but it is not intended to imply that the text of 65A(Secretariat)94 is necessarily acceptable to UK as it stands.

There is a high probability that this text could be adopted by CENELEC as a reference document for harmonization or as a European Standard. Recipients of this draft are requested to comment on the text bearing in mind this possibility.

Contents:

Clause.		Page
	Introduction	3
1	Scope	5
2	Normative References	7
3	Definitions	8
4.	Software Safety	13
5	Architecture	18
6	Software Life Cycle	21
7.	Software Requirements Specification	22
8.	Development	24
9.	Verification	26
10.	Integration	30
11.	Validation	32
12.	Documentation	34
13.	Assessment	36
14.	Quality Assurance	37
15.	Maintenance	38

Annex:

A.	Selection of Methods for Design and Assessment	39
B.	Description of Methods for Design and Assessment	54
C.	Documentation	116
D.	Rules for Development	138
E.	Software Life Cycle	147
F.	Maintenance	154

Introduction

The basic principles for the design of control systems, as applied specifically to industrial safety related applications, have been interpreted in existing standards with reference to hard-wired systems.

This International Standard has been developed to interpret these principles for safety related software in Programmable Electronic Systems (PES's). This International Standard covers software in dedicated microprocessors, programmable logic controllers (PLCs), multiprocessor distributed systems, larger scale central processor systems and other forms of safety-related PES. This International Standard also discusses the software principles and requirements and should be read in association with appropriate standards on computer hardware and system integration.

It is important to note that this International Standard establishes no additional functional requirements for safety related systems.

The principles applied in developing safety related software include:

- top-down design methods;
- modularity;
- verification of each phase;
- clear documentation;
- auditable documents;
- validation testing

Software design methods in the sense of software engineering methodology or methods of implementation are not the principal goal of this International Standard. It is explicitly stated here that good practices in software engineering and the application of quality assurance methods, for example as described in the ISO Standard 9000 series, are necessary in safety related systems. Verified modules, module libraries and strictly defined interfaces should be used. Methods to improve software ergonomics (human interface) should be applied to safety related systems, but are not covered or discussed further in this International Standard.

The current state-of-the-art is such that neither the application of quality assurance methods (so-called preventive measures) nor the application of the architectural considerations of this International Standard can guarantee the absolute safety of the system. There is no known way to prove the absence of faults in reasonably complex safety related software, especially the absence of specification and design faults. Formal methods and tools should be applied where possible, but it shall be taken into account that they do not provide a "proof of correctness".

On a superficial basis it is sometimes assumed that an operationally reliable system is automatically a safe system. A detailed analysis shows, however, that high reliability, while necessary, is not sufficient to ensure safety. Furthermore in some situations there is a tradeoff between reliability and safety as we will see later. The designer will distinguish between reliability and safety.

The extent to which reliability, safety or both have to be achieved should be agreed between the customer and the designer of the system in advance.

Reliability is oriented towards the purpose of the system and to the intended action (service); it is the extent to which a system is expected to perform the specified task. Reliability requirements are concerned with the continuous provision of the system service.

Safety looks at the causes and consequences of possible accidents. Safety requirements are concerned with making a system which does not cause accidents. The safety requirements shall ensure that the system does not reach a hazardous or unsafe state, where an event may cause an accident. Moreover it shall be transparent from the safety requirements, what actions should be taken, if an-unforeseen event in the environment leads to an unsafe state.

From the point of safety it does not matter if the system does not serve its purpose, as long as the safety requirements are not violated. (Example from the area of railway signalling: In case of a switch with a faulty sensor the green light for that track will not be set, so the train has to stop). On the other hand it is possible that the system is highly reliable, but unsafe, for example, a system with formally verified software where a safety related situation has not been specified. Furthermore, there may be a tradeoff between reliability and safety, as is illustrated by the following example.

Consider a fail-safe system with two computers operating in active redundancy. The outputs of the two computers are compared in order to detect a possible failure of one of them. After the occurrence of such a failure there are two strategies to handle the situation, depending on whether a high reliability or high safety has to be achieved:

- If a high degree of safety is required both computers have to be shut down and the process has to be brought into a safe fall back-state. No further service can be provided because there is no way to decide which of the computers is faulty. Even if there is a high probability that the faulty computer can be identified it would be impossible to detect further erroneous outputs without redundancy.
- If a high degree of reliability has to be achieved there is a probability that the faulty computer can be identified and the operation can be continued without any redundancy with the (probably) intact computer.

There is another difference between safety and reliability. Software has a reliability of its own, that is the probability of getting the intended result after submitting the input. It is possible to analyse the reliability of software packages. For software safety it is necessary to look at the whole technical system, the software must be seen in the context of the particular application. It is meaningless to speak about a safe program for sorting, if there is no hazard possible due to an erroneous sorting algorithm.

SOFTWARE FOR COMPUTERS IN THE APPLICATION OF INDUSTRIAL SAFETY-RELATED SYSTEMS

1 Scope

This International Standard specifies requirements for each stage of software generation, including design, development and operation, for computers in the application of industrial safety-related systems. Requirements for the documentation of each stage of software generation are also provided.

Additional guidance and information on how to comply with the requirements of the main part of this International Standard is given in the Annexes.

This International Standard provides requirements for each stage of software generation, including design, development and operation. Requirements for the documentation of each stage of software generation are also provided.

This International Standard is applicable exclusively to software and the interaction between software and the "wider system". Concepts or methods exist which detect and mask transient or permanent physical hardware faults. Only those concepts or methods which can be implemented in software are described.

This International Standard -

- is generically based and is applicable to both design and assessment irrespective of the application.
- allows responsible IEC Technical Committees to develop application specific recommendations.
- gives guidance on software for PESs which have safety functions where no application specific recommendations are available.
- is complementary to IEC Publication (Document being developed by the IEC/SC65A/WG10) dealing with the 'Functional safety of programmable electronic systems (PESs, defined in Clause 3) generic aspects'.
- is applicable to the software in safety related systems incorporating programmable electronics.

This International Standard is principally concerned with safety of persons, and concentrates on software to be used for safety related systems. Nevertheless it is recognised that the consequence of failure could have serious economic or environmental implications. In such cases this International Standard is applicable when assessing the protection of equipment or product.

This International Standard is applicable to safety related software required for computers used in industrial control systems. Such systems include safety actuation systems, safety system support features and protection systems. These systems may be implemented using dedicated microprocessors, programmable logic controllers, multiprocessor distributed systems, larger scale central processor systems and other forms of PES. This International Standard is applicable to the safety related software used in these systems including, for example, ladder logic application software in PLCs

Aspects for which special recommendations are made, due to the unique nature of software are:

a) considerations of the high degree of interdependence between hardware and software;

b) criteria for the choice of software, taking into account the specific characteristics of software compared with hardware;

c) the approach to software development to assure the production of the highly dependable software required;

d) the approach to software verification and computer systems validation and assessment with respect to software;

e) procedures for software maintenance, modification and configuration control.

The general task of providing guidelines for safety related PESs containing safety related software includes two aspects:

- Safety aspects within the software
- Ensuring system safety by means of software.

This International Standard provides design and assessment guidelines for safety related software and describes the currently known models of software architecture which support the safety aspects of software controlled systems. Software architecture is considered here as an essential principle in the design of software for long-term usage. The models presented here have been chosen to reflect the specific requirements of safety related systems.

2. Normative References

The following Standards contain provisions which, through reference in this text, constitute provisions of this International Standard. At the time of publication, the editions indicated where valid. All standards are subject to revision, and parties to agreements based on this International Standard are encouraged to investigate the possibility of applying the most recent editions of the standards listed below. Members of the IEC and ISO maintain registers of currently valid International Standards.

IEC 880 - 1986 Software for Computers in the Safety Systems of Nuclear Power Stations.

IEC 50 (191) - International Electrotechnical Vocabulary Chapter 191 Reliability, maintainability and quality of service (being printed).

IEC, SC65A, WG10 Draft - 1989 Functional Safety of Programmable Electronic Systems : Generic Aspects.

3. Definitions

For the purposes of this International Standard, the following definitions apply.

Architectural Design of Safety Related Software

A combination of software concepts, which under stated conditions of the system considered enhance the safety of the system by means of software. Depending on properties of the system, this includes methods and design considerations implemented in software to detect dangerous situations, to cope with these situations by returning to a fail-safe state or by introducing fault tolerant software structures.

Availability

The probability that a system will be able to perform it's designated function when required for use.

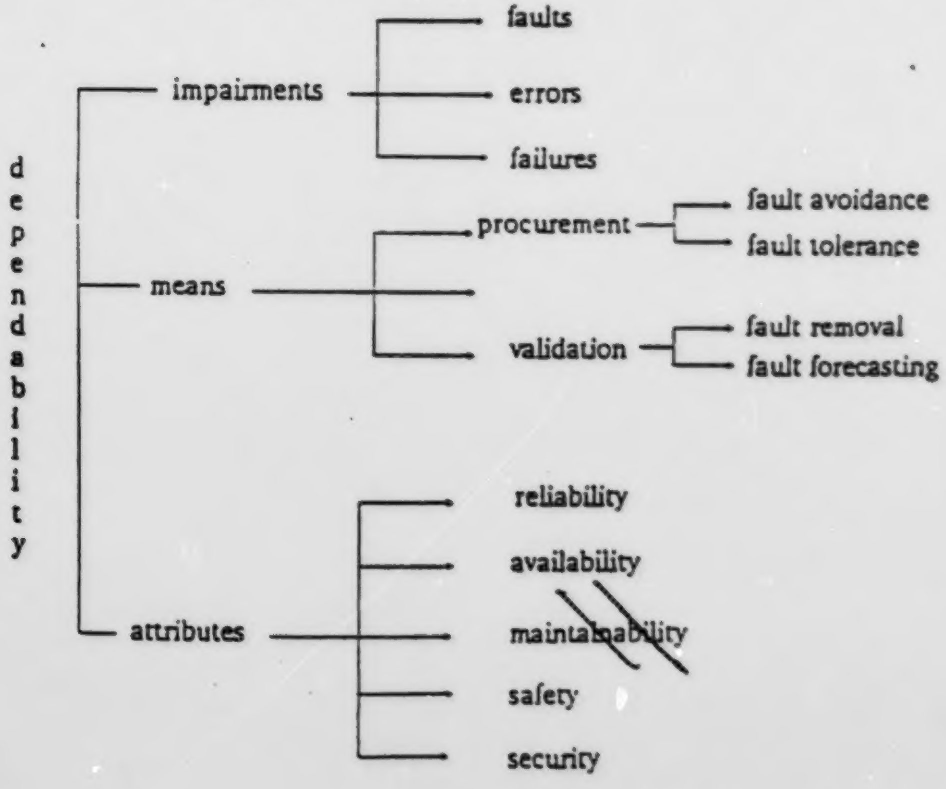
Dependability

The property of a PES that allows reliance to be justifiably placed on the service it delivers.

Dependability is a term covering all the principal characteristics known in connection with safety related software.

The connections between dependability and related terms in "safe software" are shown in the following figure 1:

Figure 1 - Dependability


dependability
impairments
faults
errors
failures
means
procurement
fault avoidance
fault tolerance
validation
fault removal
fault forecasting
attributes
reliability
availability
maintainability
safety
security

Design Fault

A design (coding, specification) fault results from a (human) failure occuring during the design of a system. A design fault causes an error, residing undetected within a (sub)system, until the input values to that (sub)system are such that the produced result does not conform to the specification. This constitutes the failure of that (sub)system. If the same inpu' values appear again, the same erroneous results will be produced.

Diversity

Existence of different means of performing a required function (e.g. other physical principles, other ways of solving the same problem).

Error

A failure occurs because the system is erroneous: an error is that part of the system state which is liable to lead to failure.

An error is, in short, a detected deviation from the agreed specification of requirements.

Fail-Safe

The built-in capability of a system such that predictable (or specified) equipment (or service) failure modes only cause system failure modes in which the system reaches and remains in a safe fall-back state.

Failure

A system failure occurs when the delivered service deviates from the specified service, where the service specification is an agreed description of the expected service.

A failure, in short, is the manifestation of an error in the system or software.

Fault

A fault is a defect that gives rise to an error.

Fault Avoidance

The use of ~~design~~ techniques which aim to prevent ~~avoid the introduction of~~ faults. ~~during the design and construction of the system.~~

Fault Recovery

The use of techniques which aim to transform the current erroneous system state into a safe state from which normal (and safe) operation can continue.

Fault Tolerance

The built-in capability of a system to provide continued correct execution, i.e. provision of service as specified, in the presence of a limited number of hardware or software faults.

Graceful Degradation

Stepwise reduction of system functions in response to detected failures while essential functions are maintained.

Maintainability

The ability of an item under given conditions to be retained in or restored to a state, in which it can perform the required function.

Mistake

A mistake is a human failure. It is a human action that produces an unintended result (in design, programming or operation).

Programmable Electronic System (PES)

A system, based on one or more computers, connected to sensors and/or actuators on a plant for the purpose of control, protection or monitoring.

Physical Fault

Physical faults are either caused by an aging effect of a component of the system or by an external interference (magnetic, electrostatic etc.). If a component becomes damaged, then the fault is a permanent fault, otherwise the fault is a transient fault.

Redundancy

Provision of additional elements or systems so that any one can perform the required function regardless of the state of operation or failure of any other. Redundancy can be implemented by identical elements (identical redundancy) or by diverse elements (diverse redundancy).

Reliability

The probability, that during a certain amount of time a system performs the functions described in the specification of requirements under the stated conditions (environment, costs, inputs and hardware, time constraints).

Risk

The combination of the frequency, or probability, and the consequence of a specified hazardous event. The concept of risk always has two elements: the frequency, or probability, with which a hazard occurs and the consequences of the hazardous event.

Robustness

The probability that a system performs the functions defined in the specification of requirements if stated conditions are not observed and/or components respective inputs are faulty.

Robustness of Software

The extent to which software can continue to operate a system correctly despite the introduction of invalid inputs and/or faults in system components.

Safe Fall-back State

A safe state which can be attained by the system at any stage in operation. The attainment of this safe state may mean that the system does not meet its specification of requirements.

Safe State

A state of a defined system in which there is no danger to human life, economics or environment under certain assumptions and specified conditions.

Safety

The expectation that a system does not, under defined conditions, lead to a state in which human life, economics or environment are endangered.

Note. For system safety, all causes of failures which lead to an unsafe state shall be included: hardware failures, software failures, failures due to electrical interference, due to human interaction and failures in the controlled object. Some of these types of failure, in particular random hardware failures, may be quantified using such measures as the failure rate in the dangerous mode of failure or the probability of the protection system failing to operate on demand. The system safety also depends on many factors which cannot be quantified but can only be considered qualitatively.

Safety Related Software

Software which ensures that a system does not endanger human life, economics or environment.

Security

Fault tolerance against deliberate interaction faults (intrusion) from internal or external sources.

Software

Intellectual creation comprising the programs, procedures, rules and any associated documentation pertaining to the operation of a data processing system. (cf ISO rules ISO/2382/1: 2nd Edition: 1984-10-1)

Note - Software is independent of the media used for transport.

Software Life Cycle

The activities occuring during a period of time that starts when software is conceived and ends when the software is no longer available for use. The software life cycle typically includes a requirements phase, development phase, test phase, integration phase, installation phase and a maintenance phase. (See Annex E)

Software Modularity

The software attribute that measures a structure of highly independent computer program units that are discrete and identifiable with respect to translating, testing and combining with other units.

System

A system is defined to consist of a set of components which interact according to a design. A component of a system can be another system (called a subsystem).

Such components (subsystems) may be, depending on the level:

- a controlling or controlled system;
- hardware, software, human interaction.

Validation

The test and evaluation of the integrated software system to ensure compliance with the safety, functional, performance and interface requirements.

Verification

The process of determining whether or not the product of each phase of the software life cycle development process fulfils all the requirements specified in the previous phase. Verification includes testing.

Notes on Failure Modes

The definitions of failure, error and fault have to be seen in the context of the system. The definitions above refer to one level of this hierarchical system decomposition.

An error is thus the manifestation of a fault in the system, and a failure is the effect of an error on the service. The failure of a system component can be considered in two different ways depending on the hierarchical level from which it is seen.

If the component itself is seen as a system (lower level view) this failure is the result of an internal erroneous state caused by a fault within the component.

On the next higher level the failure of the component is a fault possibly resulting in erroneous states and a failure of the next system level.

At first sight the definitions of fault, error and failure in this International Standard seem to deviate from the definitions given in IEV.

The reason for using these slightly different definitions lies in the fact that they refer to a clear hierarchical concept of system structure.

The connection to the definitions given in IEV may be established by recognising that the term "item", used in IEV, refers to a component of the system without regard to the level of system decomposition from which it is seen. This means that the failure of an item corresponds to the view where this component is seen as a failing (sub-) system. Afterwards, if viewed from the next higher system level, this component, or item, is faulty.

4.0 Software Safety

4.1 Software Safety Process

The process which shall be followed to ensure software safety is detailed in Figure 2 of this International Standard.

The software safety process assumes that a system hazard analysis has been carried out which conforms to the procedures detailed in the draft International Standard titled 'Functional Safety of Programmable Electronic Systems : Generic Aspects' produced by SC65A, WG10.

After the relevant documentation has been obtained then all of the safety functions allocated to the software must be identified.

Once all the safety functions within the software have been identified then a review of these functions shall be carried out. If the review team is satisfied that all the safety functions have been identified then a safety integrity level is allocated to the software (see Clauses 4.3 and 8.1). If the review team is not satisfied that all safety functions have been identified then the previous phase of identifying the safety functions is repeated.

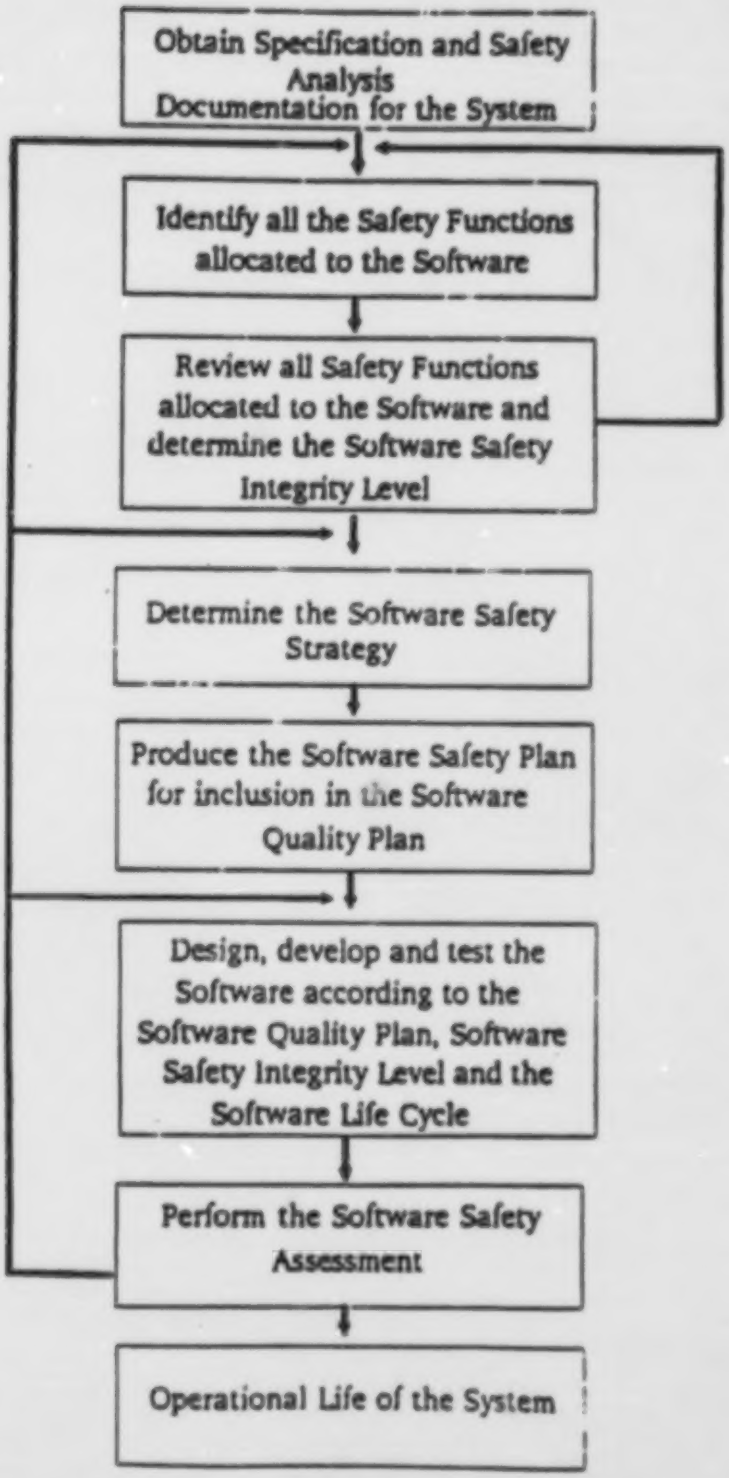
The safety strategy is now decided on. This is usually a decision between using the following competing strategies: Fault Tolerant, Fault Avoidance or both. Reference should be made to Table A1 in Annex A for guidance on this decision.

After a decision has been made on the safety strategy the safety plan may be created and incorporated into the overall Quality Plan for the software. The Quality Plan also references all of the documents produced previously detailing the rationale behind the safety integrity level and the software strategy.

The design, development and testing of the software takes place after the creation of the Quality Plan. This is followed by an assessment of the software in meeting its safety requirements as detailed in the Quality Plan. If the software does not meet these requirements then the process is returned as detailed in Figure 2 to a previous phase of the safety process.

If the safety assessment and the life cycle testing on the software are passed then the software enters its operational life phase.

Figure 2 - The Software Safety Process


Obtain Specification and Safety Analysis Documentation for the System
Identify all the Safety Functions allocated to the Software
Review all Safety Functions allocated to the Software and determine the Software Safety Integrity Level
Determine the Software Safety Strategy
Produce the Software Safety Plan for inclusion in the Software Quality Plan
Design, develop and test the Software according to the Software Quality Plan, Software Safety Integrity Level and the Software Life Cycle
Perform the Software Safety Assessment
Operational Life of the System

4.2 Software Safety Personnel

All personnel involved in design, development, assessment, installation, maintenance or management activities for safety related software shall have the appropriate training, experience and qualifications. The precise training and qualifications necessary are not defined here, and require interpretation for particular industries In all cases training, experience or qualifications are required with respect to programmable electronics specifically.

The design, development, assessment, installation and maintenance of a safety related application may require different levels or combinations of skills not necessary for other work. Particular consideration must therefore be given to staff selection, training and responsibility. There will be a need, in most cases, for a reserve of suitable personnel. Refresher training may be required in order to ensure that personnel are familiar with all aspects of the safety related system. This is particularly important where an operator is expected, in the event of a system shutdown, to maintain safety by manual or other means. In such circumstances adequate operating procedures and instructions are particularly important.

The specialised nature of safety related technology can impede communication between those dealing with the safety related system and others. This must not prevent the proper monitoring of work, associated with the safety related system, necessary to ensure that safe practices are adhered to. Safety and other personnel shall therefore be trained or otherwise familiarised with the safety related system including relevant operation and maintenance features

4.3 Software Safety Integrity Levels

The required safety integrity level for the software shall be decided on the basis of the level of risk associated with the software. There is not necessarily a direct relationship between the level of risk associated with a system and the level of risk associated with the software. Other non-software components affect the level of risk associated with the software.

Risk is defined as 'the combination of the frequency, or probability, and the consequence of a specified hazardous event'. The concept of risk always has two elements: the frequency, or probability, with which a hazard occurs and the consequences of the hazardous event.

In any particular application there will be a level of risk that can be deemed 'tolerable'. For a situation where the level of risk, without precautions, is not tolerable, there is a need to ensure that the applied safety related systems reduce the risk to an acceptable level.

Risks which shall be taken into account are those associated with the following consequences of hazards:

- loss of human life or lives,
- injuries to or illness of persons,
- environmental pollution,
- loss of or damage to property.

Risk may be specified as a quantified probability or a combination of quantified probabilities but it is not possible to specify the software safety integrity in the same manner.

Software safety integrity is specified as one or more of five levels;

Level	Description of Software Safety Integrity
1	Very High
2	High
3	Medium
4	Low
5	Normal

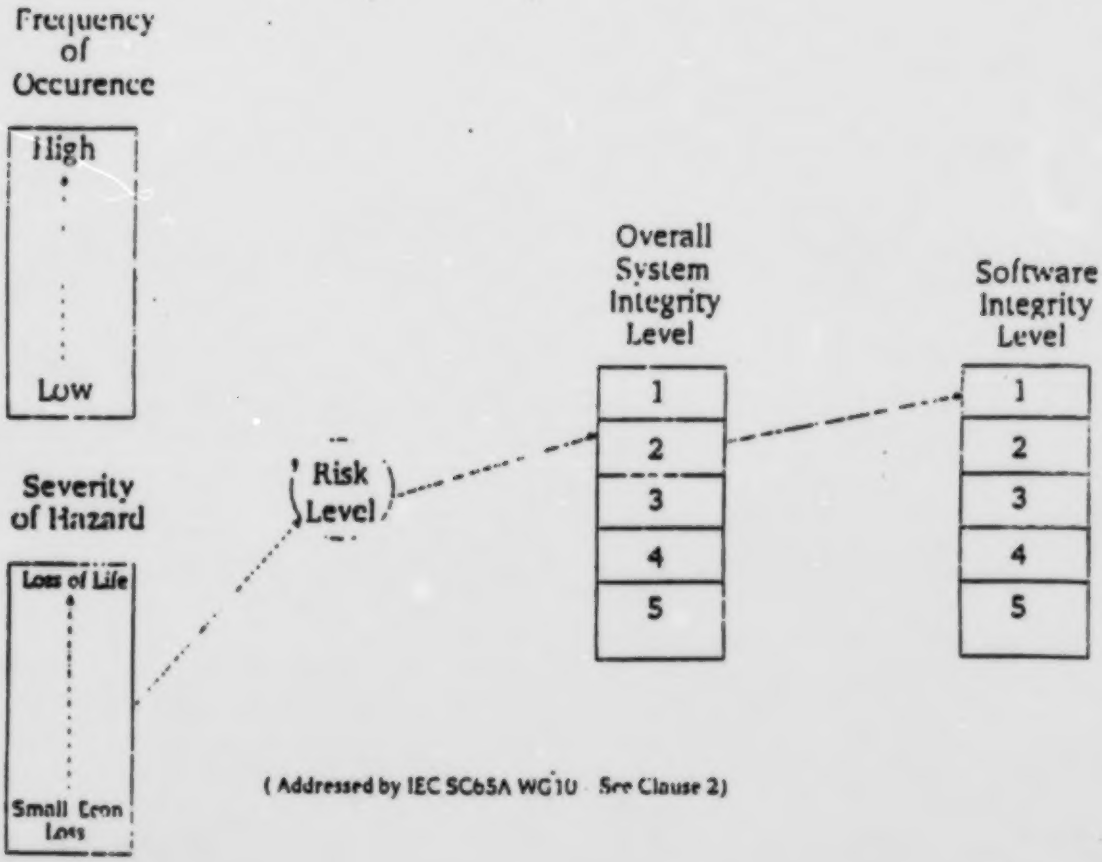
Compliance with ISO 9000 is mandatory at all levels of Software Safety Integrity. The base level, Normal, represents a system where compliance to ISO 9000 is sufficient.

4.4 Guidelines on Software Safety Integrity and Risk

Clause 4.3 details five levels of Software Safety Integrity and the four hazards that this standard is concerned with. There is no state of the art method of mapping the frequency of occurence and the hazard to the Software Safety Integrity without also considering the risk level to be tolerated and the Overall System Integrity. The tolerable risk level and the Overall System Integrity are dealt with by the standard being prepared by IEC SC65A WG10 (See Clause 2)

To determine the Software Safety Integrity it is necessary to identify the frequency of occurence and the severity of hazard which combine to form the risk. As there will be a level of risk which is considered tolerable this is determined as a step to assessing the Overall System Integrity (which will consider such issues as configuration). From the Overall System Integrity the Software Safety Integrity can be determined. An illustration of this mapping is shown in Figure 3.

Figure 3 - Possible mapping between Hazard Consequences and Safety Integrity Levels


Frequency of Occurence
High
Low
Severity of Hazard
Loss of Life
Small Econ Loss
Risk Level
Overall System Integrity Level
1
2
3
4
5
Software Integrity Level
1
2
3
4
5
(Addressed by IEC SC65A WG10 See Clause 2)

5. Architecture

5.1 General

Complexity is a major cause of design faults. A general rule for the design of safety related systems shall be to minimise the safety related part of the application and to ensure that the safety related part is independent of the other parts of the system. In particular, critical and non-critical parts should be separated in some way.

There are different approaches to the design of safety related software, such as Fault Tolerance and Fault Avoidance. Furthermore different types of configurations can be used to improve the system, such as Diversity and Redundancy.

5.2 System Design Approaches

There are two different approaches to the design of safety related software, these are Fault Avoidance and Fault Tolerance which are detailed in Clauses 5.2.1 and 5.2.2 respectively.

5.2.1 Fault Avoidence

In addition

Fault avoidance is constructing the software such that it contains a minimum number of faults (errors). This may be achieved by keeping track of all possible effects that any statement in the software may ever have on the data space. ~~Another way~~ is at the end of the development process to verify using methods appropriate for the Safety Integrity Level, no faults (errors) can be detected.

A set of tables in Annex A details which techniques are applicable.

5.2.2 Fault Tolerance

Three basic types of safety related fault tolerant software system are recognised by this International Standard.

(a) Fail-safe systems

A prerequisite for a fail-safe system is that there exists at least one safe fall-back state, which can be reached not only under all operational conditions, but also, and especially, after the manifestation and detection of a failure in the system. In order for the system to be fail- safe, the set of states from which the safe fall-back state can be reached shall include not only the "normal" operational states of the system, but also other intermediate, and possibly unsafe, states due to system failure. Fail-safe systems shall make use of this safe fall-back state in case of failure.

(b) Fault compensating systems

In a fault compensating system there is no safe fall- back state, or the system does not for some reason make use of a safe fall-back state which does exist. The system may not make use of the safe fall-back state in order to improve reliability or improve functionality. The system shall cope with faults (for example by redundancy, fault detection and decision algorithms, fault correction, reconfiguration) in such a way that the system safety is preserved, along with the continued provision of the whole, or part, of the specified functionality.

(c) Fault Masking

The aim of fault masking is to continue the operation of the system without undue interruption in the event of a fault occurring.

Fault masking means detecting that a failure has occurred and then taking corrective action. When faults are masked properly they appear to have little or no effect on the operation of the system.

There are two categories of fault masking; detecting and correcting the effect of a fault and physical masking (two out of three votes).

Guidelines for the implementation of fault tolerant systems are included in Annex D, Rules for Development.

These types of fault tolerant systems can be combined together to produce mixed mode systems. A mixed mode system is one which is partially fail-safe or has a temporary safe fall-back state. Two types of mixed mode systems are detailed below.

(a) Fail-safe with residual functions (partially fail-safe)

In the safe fall-back state some minimal functionality (for example a cooling system) is be maintained in order to preserve safety.

(b) Temporary failsafe systems

There exist circumstances, under which the system either has no safe fall-back state, or the safe fall-back state cannot be reached.

Fault tolerance can also be obtained by the use of high level strategies which cope with the inadequacy of the model of the real world used. High level strategies include methods like background simulation (concurrent simulation) in flexible manufacturing systems or the use of expert systems to forecast events and reactions of systems and to take countermeasures in case of critical deviations of the system.

These high level strategies achieve two main goals allowing a further differentiation of methods:

(a) Prevention of dangerous events,

(b) Provision of preventative countermeasures

5.3 Safety Integrity Levels and System Design Strategies

Once the safety integrity level for the software has been assigned (see Clause 4) then Fault Tolerant and Fault Avoidance design strategies can be adopted. Both strategies should be present at all levels of safety integrity. However the selection of a particular technique will vary with the level of safety integrity. For example at low levels of integrity a suitable fault avoidance technique for modelling the software specification might be data flow diagrams, whilst at the higher levels formal mathematical methods may be more appropriate. Tables and details of the relative ranking of fault avoidance and fault tolerant techniques are given in Annex A.

5.4 Methods to Improve Safety

5.4.1 Diversity

The safety of a system can be improved by methods which decrease the number of undetected design faults.

The main idea of diversity is to design and implement a system more than once in different ways, feed the different versions of the system with the same data and compare the results. If any difference in the time domain or in the value domain is detected, a design fault may be present and should be anticipated.

For a more detailed description of diversity see Annex B, Clause B.1.2.

5.4.2 Redundancy

Redundancy (multiple application of identically designed components) increases the reliability of a system. It is a method for detecting or tolerating physical faults. Redundancy is an important method for the design of failsafe systems since reliability is closely related to safety.

For a more detailed description of redundancy see Annex B, Clause B.1.1.

6. Software Life Cycle

Any project will normally be divided up into a number of phases. The phases of the life-cycle are detailed in Annex E, Software Life Cycle, of this International Standard. Each phase is self-contained but will depend on other phases and will, in its turn, be depended on by others. These phases are recognisable by the specific activities pertinent to them.

For safety related applications, these phases shall be formalised and none of the identified phases of the life-cycle shall be omitted.

The following general requirements apply to all software being developed as part of the project;

- The whole software life cycle shall be considered.

- Each phase of the software life cycle shall be divided into elementary tasks with a well-defined activity for each of them.

- Quality Assurance Procedures shall be run in parallel with the other tasks of the life cycle.

- Each phase shall include generation of the appropriate documents (see Annex C, Documentation, and Annex E, Software Life Cycle). A list of documents required as a minimum through the software life cycle is given in Clause 12.2.

- Each phase shall be systematically terminated by a verification or review. The software design shall be terminated by the Software Design Review (see Clause 9.3).

- Every verification or review shall result in a report on the analysis performed, the conclusions reached and the resolutions agreed. This report shall be included in the documentation list.

- As many components as possible that are qualified through operation already shall be used.

7. Software Requirements Specification

The Software Requirements Specification shall be derived from requirements of the Quality Plan, in particular the Software Safety Plan and the System Specification and is part of phase 3 of the software life cycle (see Clause 6 and Annex E). The Software Requirements Specification may either be a separate document or part of the System Specification. The System Specification is detailed in the draft International Standard titled 'Functional Safety of Programmable Electronic Systems : Generic Aspects' produced by SC65A, WG10 and is a description of the combined hardware/software systems and states the objectives and functions assigned to the system.

The software requirements shall be the properties of the product, not the project. They shall describe what has to be done and not how it has to be done. An acceptable approach to the development and content of the Software Requirements Specification, which is consistent with this Clause, is given in Annex C

Whereas the main purpose of the Software Requirements Specification is to form the basis for software development, the licensing aspects should not be neglected. Therefore, it may contain aspects of minor importance to software design which however, provide a background for licensing. Such aspects may be:

- recommendations for functions or engineered safety features,
- other items that provide the background for specific requirements.

The Software Requirements Specification shall include the software functional requirements, the software reliability requirements and the software safety requirements.

The software functional requirements represent an expansion of the functions which are assigned to the system and which are to be implemented by the system.

The software should be able to collect automatically all information gained during periodic testing.

Software functional requirements shall be presented according to a standard whose formality should not preclude readability. The requirements shall be unequivocal, testable or verifiable, and feasible.

The software reliability requirements represent an expansion of the reliability requirements of the system. They shall be derived in a similar way to the functional requirements.

The interface between the system under consideration and any other system either within or outside the plant, wherever a direct connection exists or is planned, shall be identified and documented indicating the specific interfaces and the related software requirements.

The constraints between hardware and software shall be described. The Hardware Requirements Specification shall be referenced.

Special operating conditions such as plant commissioning, repair and maintenance shall be described.

The software shall monitor both itself and the hardware. This is considered a primary factor in the achievement of the overall system reliability requirements.

Self-supervision shall meet the following requirements unless it is proved that other means provide the same degree of safety:

- Those parts of the memory that contain code or invariable data shall be monitored to prevent any changes.
- The process shall be monitored for logical correctness of function.

The software shall be designed in such a manner that all essential functions detailed in the software safety requirements shall be testable during the operation of the plant.

If any failure is detected during plant operation, appropriate and timely automatic actions shall be taken. This may require giving due consideration to avoiding spurious failures.

System self-checking shall be designed so as to meet the requirements of periodic testing. There shall be periodic testing to ensure that:

a) Every single function is coverable by periodic testing.

b) Any degradation of the execution of safety functions shall be detected.

c) The basic self-checking functions shall be tested during periodic tests.

The use of a formal specification technique or language may help to show coherence and completeness of the software functional requirements. If the formal specification is only readable by persons with special training then a more readable interpretation shall be provided as well. Automatic tools may be used to assist in generating the specification.

The Software Requirements Specification shall be provided for:

- assessment and approval prior to design.
- approval with respect to feasibility and readability.
- assessing compliance with the overall plant safety requirements.
- assessing the compliance of the software with its specification.

8. Design and Development

The software design and development phases shall be divided into the following phases:

- architectural design,
- detailed design,
- coding,
- software integration.

During the architectural design phase the basic safety concepts shall be selected and implemented according to clauses 4 and 5.

The Software Requirements Specification shall be available, before the design and coding phase of program development begins.

8.1 Principles for Development (design and coding)

The principles detailed in this Clause are intended to enable software to be produced which contains a minimum number of errors and shall be readily verifiable.

The following general principles for development are based on experience in producing software.

- The software design shall include self-monitoring of control flow and data movements. On failure detection appropriate action shall be taken.

- The program structure shall be based on a decomposition into modules. Each module shall have its own Software Module Specification and Software Module Test Specification.

- All software coding shall be treated, in all respects, as safety related unless non-safety related sections have been determined. With some systems it may be possible to clearly establish that specific sections of the software coding (considered as non-safety related) do not affect the safety functions.

- All software coding shall be treated as safety related for those situations where there are safety functions of differing levels of safety integrity. It may be possible to clearly establish that specific software coding sections belong to a lower integrity level and do not affect the safety functions of the higher integrity levels. Unless this is the case all sections shall be treated in all respects as software coding belonging to the highest integrity level existing in the safety related system in question.

  Note: Normally such independence between different software sections can only be established by means of a detailed assessment of the hardware and software of the safety related system.

- Each module shall be readable and comprehensible from start to end. Its function shall be understandable on the basis of its interface.

- Good documentation shall be provided (see Clause 12).

From these principles the following recommendations are derived:

a) Methods of obtaining the required reliability including self-monitoring should be chosen at the outset of the development.

b) A top down approach to software development is preferred. A top down approach to software development should be supplemented by a bottom up approach to certain parts of the software such as device drivers etc.

c) A conceptual model of the software system structure should be adopted at the beginning of each software project and only modified under appropriate change controls.

d) Each program and module shall be written with the aim of allowing easy testing.

e) Where standard software from a manufacturer or supplier is used, it shall either be shown to have operated satisfactorily or shall have been subject to the same verification and validation procedure from the coding phase on, as for any newly developed software.

f) Existing verified components should be reused where possible.

8.2 Language and Translator

Detailed guidance for selection of language, translator, etc. is given in Annex D.

The following shall be considered as common basic rules for the selection of a programming language for use in a safety related software system.

Programming languages with a validated translator or one which has been thoroughly tested shall be used. A validated translator is one which has a 'Certificate of Validation' to a recognised National or International language standard. Where no thoroughly tested or validated translator is employed, additional verification shall show that the result of the translation is correct.

Either the programming language shall be completely and unambiguously defined or the use of the programming language shall be resticted to completely and unambiguously defined features. In a similar way where there is doubt about the correct translation of a specific language feature or a particular combination of such features then the use of these features shall be proscribed. Such restrictions shall, where possible, be enforced by automated means.

Problem oriented languages are strongly preferred to machine oriented languages, for example ladder logic rather than assembler.

Coding standards shall be used for all safety related software. These coding standards shall specify good programming practice and shall proscribe unsafe language features, examples of which are given in Annex D, Clause D.2.

Where available automatic testing tools should be used, for example debuggers.
Where available integrated development tools should be used. These include structural analysis and design tools.

9. Verification

9.1 Verification Plan

Concurrent with each phase of the software development life cycle a software verification plan shall be established. The plan shall document all the criteria, the techniques and tools to be utilised in the verification process for that phase. It shall describe the activities to be performed to evaluate each item of software and each phase to show whether the system functional, reliability, performance and safety requirements are met. The level of detail shall be such that an independent group can execute the verification plan and reach an objective judgement on whether or not the software is adequate.

The requirement for an independent verification group implies that verification shall be carried out either by an individual or an organisation which is itself independent of the individual or organisation developing the software. The appropriate verification independence depends on the safety integrity level. The extent to which independence is required is detailed in Clause 13 and guidance is given in Annex A, table A2.

The verification plan shall be prepared by a verification team addressing the following:

a) selection of verification strategies; systematic, random or both, with test case selection according to required functions, special features of program structure, or both, see Annex A;

b) selection and utilisation of the software test equipment;

c) selection and documentation of verification activities;

d) evaluation of verification results gained from verification equipment directly and from tests;

e) evaluation of whether the reliability requirements are met.

After each verification there shall be a verification report produced stating either that the software has passed the verification or that the software has failed the verification with the reasons that the software failed included.

9.2 Design Verification

9.2.1 Software Requirements Verification

Once the Software Requirements Specification has been established, and before the next phase, Software Design, begins, verification shall address the adequacy of:

a. The Software Requirements Specification in fulfilling the requirements (especially the safety requirements) assigned to the software by the System Specification.

b. The Software Acceptance Test Specification as a test of the Software Requirements Specification.

Furthermore, verification shall check for incompatibilities between:

a. The Software Requirements Specification and the System Specification.

b. The Software Requirements Specification and the Software Acceptance Test Specification.

c. The Software Acceptance Test Specification and the System Integration Test Specification.

9.2.2 Software Design Verification

After the Software Design Specification has been established and before the next phase, Software Module Design, begins, verification shall address the adequacy of:

a. The Software Design Specification in fulfilling the Software Requirements Specification.

b. The Software Integration Test Specification as a test of the Software Design Specification.

Furthermore, verification shall check for incompatibilities between:

a. The Software Design Specification and the Software Requirements Specification.

b. The Software Design Specification and the Software Integration Test Specification.

c. The Software Integration Test Specification and the Software Acceptance Test Specification.

9.2.3 Software Module Verification

After the Software Module Specification has been established and before the coding begins, verification shall address the adequacy of:

a. The Software Module Specification in fulfilling the Software Design Specification.

b. The Software Module Test Specification as a test of the Software Module Specification.

Furthermore, verification shall check for incompatibilities between:

a. The Software Module Specification and the Software Design Specification.

b. The Software Module Specification and the Software Module Test Specification.

c. The Software Module Test Specification and the Software Integration Test Specification.

9.3 Software Design Review

The completion of the verification of the Software Module Specification is an important milestone in the development of the software system. This milestone signals the end of the software design and enables the software coding to commence.

Before the software coding can commence a Software Design Review shall take place.

The Software Design Review shall be conducted by an independent third party, as identified in table A2, Annex A, and shall address the following:

a) the adequacy of the Software Requirements Specification for meeting the requirements set out in the Quality Plan for safety and in the System Specification for functionality, reliability, performance and safety.

b) the adequacy of the Software Design Specification for the Software Requirements Specification with respect to consistency and completeness down to and including the module level;

c) the decomposition of the Software Design Specification into software modules and the Software Module Specifications with reference to:

- feasibility of the performance required;
- testability for further verification;
- readability by the development and verification team;
- maintainability to permit further evolution;

d) the degree to which quality requirements are met.

e) the adequacy and results of the previous software verification plans.

The result of the Software Design Review shall be documented in a Software Design Review Report. This report shall include

- items which do not conform to the Software Requirements Specification, Software Design Specification or Software Module Specifications.

- items which do not conform to the design standards;

- items which do not conform to the Quality Assurance Procedures;

modules, data, structures and algorithms poorly adapted to the problem.

9.4 Coding Phase Review and Testing

The code review and testing activities begin with code reviews which should as far as possible be supported with analysing tools. The modules are then tested individually. As the modules are integrated together, testing continues, following a bottom up strategy until the complete software has been tested as a whole.

At the module level, the software is not yet integrated into the system, therefore it can be extensively tested. Each module shall have a Software Module Test Specification which the module shall be tested against. The Software Module Test Specification is developed, concurrently, with the Software Module Specification. The purpose of module testing is to show that each module both performs its intended function and does not perform unintended functions.

9.5 Software Integration Review and Testing

After the integration of all software modules, the software shall be tested against the Software Integration Specification. The Software Integration Test Specification is developed, in parallel, with the Software Design Specification. This test shall show that all modules interact correctly to perform their intended function and do not perform unintended functions.

9.6 Software Acceptance Review and Testing

After the software system has satisfied the Software Integration Test Specification, it shall be tested against the Software Acceptance Test Specification. The Software Acceptance Test Specification is developed in parallel with the Software Requirements Specification. This test shall show that all of the requirements in the Software Requirements Specification are correctly performed and the software system does not perform unintended functions (see Clause 11).

10. Integration

Integration is the process achieved by the developer to verify that the system in parts and the system as a whole operate together according to the System Requirements Specification. When developing a complex system it is impractical to develop the individual modules and then assemble the complete system in one step to verify the function. The verification and test process is normally carried out in stages. Each module is individually tested by the developer. The modules are then integrated into sub-systems in stages and verified. The sub-systems are then combined and tested in the same manner into the complete system.

The system should be tested in an environment as similar as possible to that in which it will be operated.

For a Programmable Electronic System (PES) one can normally identify the following main integration steps;

- software system integration,
- hardware system integration,
- total PES integration,
- total process and PES integration.
  (Installation and Commissioning)

These integration tests can be combined. In some instances the final integration test is combined with the ~~formal~~ validation test.

Special attention should be given to the following aspects of testing:

- The integration test should be thoroughly planned.
- The test results should be recorded to the level defined in the integration plan.
- Any discovered problems and their possible impacts on any other parts of the software should be noted, together with the personnel responsible, in order to allow the problems to be traced until they are solved.
- Areas impacted by any modifications should be identified and retested.
- Test adequacy.
- The safety related parts.
- The hardware and software configuration during tests.

The specific procedures necessary for the integration depend on the nature of the system design and cannot be included in this standard. However, such procedures shall be established, and bedocumented by the integration test plans and specifications. The procedures shall be reviewed and the documents shall be placed under configuration management and change control.

The documentation shall cover the following activities;

- the split of the system into integration levels,
- test cases and test data,
- types of tests to be performed, for example functional test, performance test, usability test,
- test environment, tools and programs,
- test criteria on which the completion of the test will be judged.

The integration plan shall be prepared as early as possible in the development process to allow any integration requirement to be included in the design of the hardware or software and also to develop special integration tools.

The records from the integration test shall be analysed, reviewed and approved.

11. Validation

11.1 Purpose

Validation is the process of ensuring that the complete system fulfils its requirements. For a software system such validation will be performed as part of an overall hardware/software acceptance exercise. Validation is closely connected to the Assessment of the system and to the Integrated System Verification. Validation demonstrates the conformance of the hardware and software as a whole with the System Requirements and in particular the Safety System Requirements.

Normally validation commences after the System Integration phase. If the software or any other system part has been modified after any approved validation, then revalidation is necessary. During revalidation only those parts of the system that have not been affected by any of the modifications may be excluded. The exclusion of any parts of the system from revalidation shall be justified in the validation report for the revalidation.

11.2 Tests and the Validation Plan

Testing shall be the main validation activity. It shall be conducted in accordance with the Validation Plan. The plan shall identify the validation for static and dynamic cases.

The system shall be exercised by static or dynamic simulation of input signals present during normal operation, anticipated occurrences and accident conditions requiring system action. Each safety function shall be confirmed by representative tests.

The tests performed shall be in accordance with the mandatory and recommended techniques table in Annex A.

It is recommended that the tests:

- cover all signal ranges, and the ranges of computed or calculated parameters in a fully representative manner;
- cover the logic combinations comprehensively;
- cover timing requirements in a representative manner;
- ensure that accuracy and response times are confirmed;
- provide all typical displays to the operators;
- provide for all typical sequences of output signals;
- ensure that correct action is taken for any equipment failure or failure combination stated in the requirements.

The Validation Plan shall state for each required function or other requirements:

- required input signals with their sequences and their values,
- anticipated output signals with their sequences and their values,
- acceptance criteria,

The Validation Plan shall be developed, the validation activities carried out and the validation results evaluated by an independent validation team having not participated in the software implementation of the requirements (design and coding).

The Validation Plan shall be agreed with the assessor of the system or the party representing the assessor. This agreement shall also make statements concerning the presence of the assessor during testing.

Equipment used for validation shall be calibrated appropriately. Hardware and software tools used for this validation need no special verification. They should, however, be shown to be suitable for the purpose.

11.3 Probabilistic Aspects

It may be agreed with the assessor to conduct an extensive validation phase instead of the extensive verification phase. In this case a large number of test cases shall be required and the results of the validation plan shall be evaluated probabilistically.

It may exceptionally be agreed with the assessor to accept operating experience in other safety related or non safety related applications as a substitute for some part of the verification or validation activities. In some cases such experience shall be required. Any experience shall be evaluated according to probability theory.

Probabilistic techniques shall be applied only when personnel with the necessary mathematical background are available. They are recommended, when the necessary tools or data are already available.

11.4 Validation Report

The results of the validation shall be documented in the Validation Report. This report shall identify the hardware and software used, the equipment used and its calibration, the simulation models used, and any discrepancies and corrective actions according to the procedures detailed in the related documents.

The report shall summarise the results of the validation and shall assess the system compliance with all requirements.

The results shall be retained in a form that is auditable by persons independent of the validation. Software tools used in the validation process shall be identified as an item in the Validation Report. Simulations of the plant and its systems used for the validation shall be documented.

12. Documentation

12.1 General

During software development the end of each design phase shall be marked by production of a set of formal documents: a specification and a test specification. A formal verification then takes place to assure the completeness and the adequacy of the documents. The verification shall be performed according to the Software Quality Plan and produces a Verification Report

The documents should be structured to allow continued expansion and reification in parallel with the design process.

The aim is to produce an integrated set of documents which allow for the audit of the design and development process. Each document shall have a defined relationship to the other documents.

Styles of documentation should be selected according to the specific purpose of the document. Appropriate styles include:

- narrative description
- arithmetic expressions
- algorithm descriptions in pseudo code
- diagrams and drawings

It is preferable to select graphical representations whenever possible. The documentation itself shall follow appropriate National and International Standards. The higher levels of documentation (Software Requirements Specification) should be understandable to project management and plant specialists, not to software specialists alone.

If the safety related software is less than 10 pages of source code then the documentation for the preliminary and detailed design phases may be combined on the agreement of the assessor.

Decided this wasn't a good idea.

12.2 Documentation List

Software Quality Plan

Software Requirements Phase
- Software Requirements Specification
- Software Acceptance Test Specification
- Software Requirements Verification Plan
- Software Requirements Verification Report
- Software Validation Plan

Software Design Phase
- Software Design Specification
- Software Integration Test Specification
- Software Design Verification Plan
- Software Design Verification Report

Software Module Design Phase
- Software Module Specification
- Software Module Test Specification
- Software Module Verification Plan
- Software Module Verification Report

Software Design Review Phase
    Software Design Review Plan
    Software Design Review Report

Code Phase
    Source Code and supporting documentation

Software Testing Phase
    Software Module Test Report
    Software Integration Test Report
    Software Validation Report

Software Acceptance Phase
    Software Acceptance Test Report

12.3 Details Of Documents

All documents listed in the previous Clause are described in detail in Annex C, Documentation.

13. Assessment

Assessment is the process of taking a lateral look at the whole of the development process to check that the system is fit for its intended application purpose

All safety related software shall be subject to assessment by a person/team designated as the Assessor. The qualifications and experience of the Assessor shall be such that they are demonstrably suited to the task.

The assessment shall be undertaken by persons independent from the design team and can range from in-house to external third party dependent upon the level of safety integrity sought for the software.

The degree of independence in particular situations will depend upon a number of factors including;

- the safety integrity level,
- company expertise,
- company management structure,
- legal or contractual requirements.

It is not possible at the generic level of this International Standard to give more specific recommendations, but the higher the required safety integrity level then the greater the degree of independence in the assessment. The Table A3, in Annex A, is intended to give guidance on the degree of independence required for a particular safety integrity level

The approach to the assessment of safety related software is closely related to the approach to the development of safety related software.

The choice of architecture and methods for the development of software is described in Clauses 5 and Annex A of this International Standard. The safety integrity level of the software is the basis on which architecture, methods for development, assessors and methods for assessment are chosen.

The table A2 in Annex A is intended to give guidance on the areas which the assessment should address for each safety integrity level.

Assessment is used to establish the integrity of the product and is a superset of validation. The software shall therefore beassessed against a wide variety of safety issues with respect to the system.

14. Quality Assurance

The supplier shall have and shall use a Quality System according to ISO 9000 standard series and implement the relevant part of these standards for the software product.

The supplier shall prepare and document a Quality Plan to implement the quality activities on the basis of the quality system, and make it understood and observed by the organisations involved in the development of a safety related system.

The Quality Plan shall always be updated along with the progress of development, and items concerned with each phase shall be completely defined before starting that phase.

The Quality Plan shall be formally reviewed and agreed by all organisations concerned in its implementation.

The following items shall be specified or referenced in the Quality Plan:

- quality objectives including,
  - means of achievement,
  - means of evaluating the degree of achievement,
- definition of each phase including,
  - entry and exit criteria,
  - inputs and outputs of each phase,
  - major quality activities in each phase,
  - organisational unit responsible for each activity,
- system integration procedures,
- testing and validation plans,
- special items anticipated to affect quality,
- maintenance plans,
- development plan,
- reviews and assessment activities,
- configuration management.

The configuration activities shall include procedures for identification of all software items and their documentation during all phases. Each individual software item shall have a unique identification. Each version of a software item shall be identified.

The supplier shall establish and maintain procedures to record, manage and report on the status of software items.

For released products there shall be a procedure to facilitate traceability of the software product. This facility shall be used during the quality audit of a software product.

15. Maintenance

The maintenance of software is intrinsically different from the maintenance of hardware. Software maintenance involves either the correction of design faults, the enhancement of the design in response to additional requirements or the adaption of the design in response to changing requirements.

The complexity of such systems makes it difficult to test the system in all conceivable conditions. Faults introduced during the development phase remain in a system despite of careful testing or can be introduced during the maintenance work. It is therefore important that all maintenance is carried out using carefully planned and regulated procedures.

Changes to systems may be categorised as:

- Corrective maintenance in response to faults and anomalies.
- Perfective maintenance, brought about by the need to simplify or improve the system.
- Adaptive maintenance, whereby the system changes to reflect changes in the environment.

The maintenance task in safety related systems is often more complex because of the impact of the change within the system on other related (sub)systems. Consideration of the environment and the influence of safety and regulatory bodies also need to be considered. Personnel are frequently required to work under pressure, process/plant constraints and regulatory directives. Maintenance, therefore, needs expertise, automated tools and careful planning and management.

The standards specify techniques, procedures and policies applicable to both the hardware and software aspects of safety related computer systems. However, this International Standard concentrates on the software aspect because many aspects of the hardware maintenance are well founded, whereas the art of software maintenance is a younger discipline.

Many of the requirements for software maintenance are fulfilled by treating each maintenance task as a mini project, applying the methods and procedures identified elsewhere in this International Standard.

There are two aspects where special considerations need to be given to maintenance:

a. control of error reporting, change authorisation and software/system configuration.

b. testing, verification, validation and assessment.

[INFORMATIVE]

ANNEX A

Selection of Methods for Design and Assessment Of Safety Related Software

A.1 Introduction

This Annex is not definitive, but is included to illustrate how a software integrity level approach can be developed.

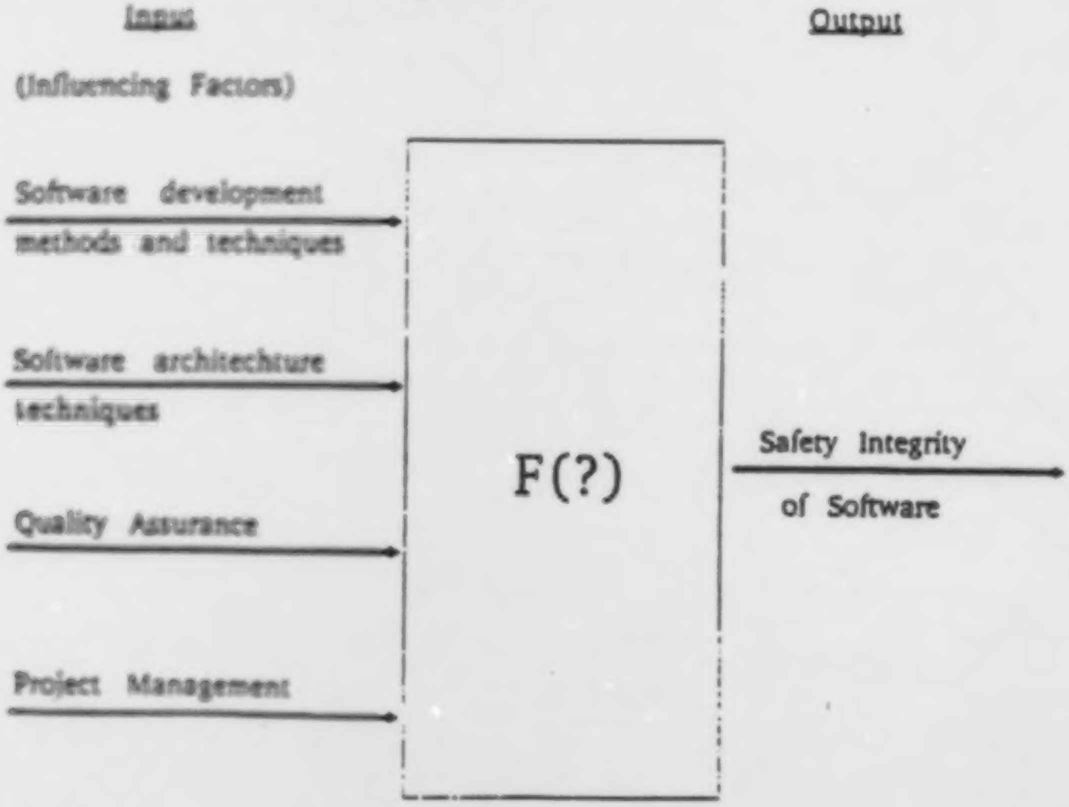
For each application a 'package of techniques' is to be constructed, which are relevant to the Safety Integrity Level assigned to the software. It should be stressed that the selection of a 'package of techniques' from these tables does not in itself ensure that the software is safe for that integrity level.

A.2 Safety Integrity Levels

As indicated in the introduction to this International Standard it is not possible to prove the absence of faults in complex safety related software, nor is it possible by testing to measure the softwares reliability or integrity. The approach taken in this International Standard to overcome these problems is to give recommendations about various approaches to the software development process with the aim of providing confidence that the software is suitable for a given level of integrity. Clearly the higher the integrity requirements the more onerous the steps involved in developing the software.

Conceptually the process of producing safety related software can be considered by reference to Figure A1 below. It is not possible to measure the output (safety integrity) as the current state of the art therefore concentrates on identifying the key inputs or influencing factors for achieving software safety integrity.

Figure A1


Input
(Influencing Factors)
Output
Software development methods and techniques
Software architechture techniques
Quality Assurance
Project Management
F(?)
Safety Integrity of Software

The next step in the development of this approach is to characterise or find attributes of the influencing factors, for example rigour of notation is an attribute of the software development process, and to then find measures on and criteria across the safety integrity levels recommended in this International Standard.

Figure A2

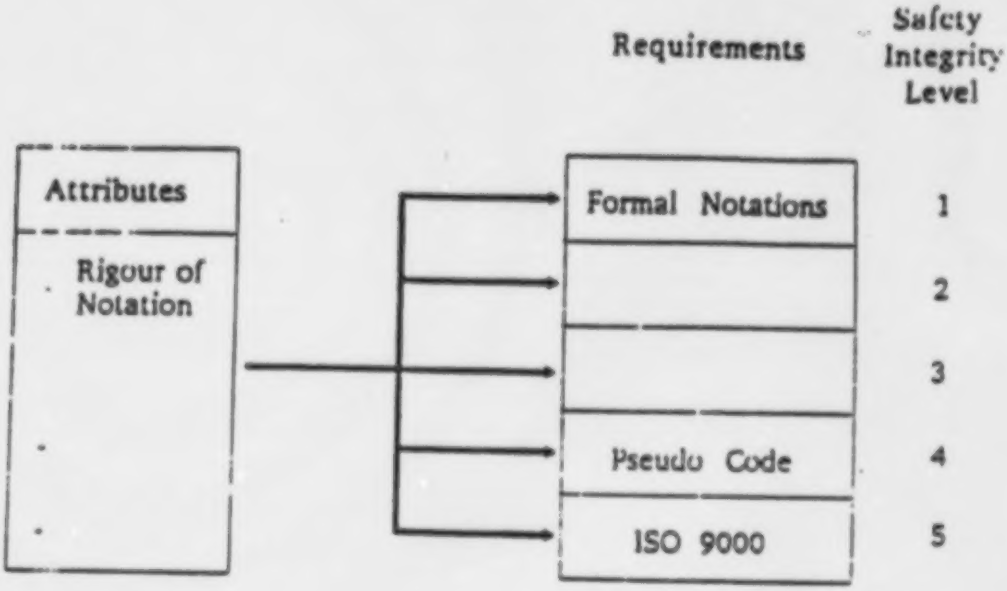

Requirements
Safety Integrity Level
Attributes
Rigour of Notation
Formal Notations
1
2
3
Pseudo Code
4
ISO 9000
5

Figure A2 gives an indication of how this approach works, but unfortunately at this stage of knowledge it is not possible to fully achieve this.

As a first step this International Standard has given a relative ranking of the main software development and safety techniques. Whilst this does not fully meet the attribute-based approach indicated above, it does indicate those techniques which are more likely to be required at the high integrity levels.

A.3 Use of Tables

The tables detailed in this Annex are intended to be a design aid to the production of safety related software.

Figure A3 - Structure of Tables

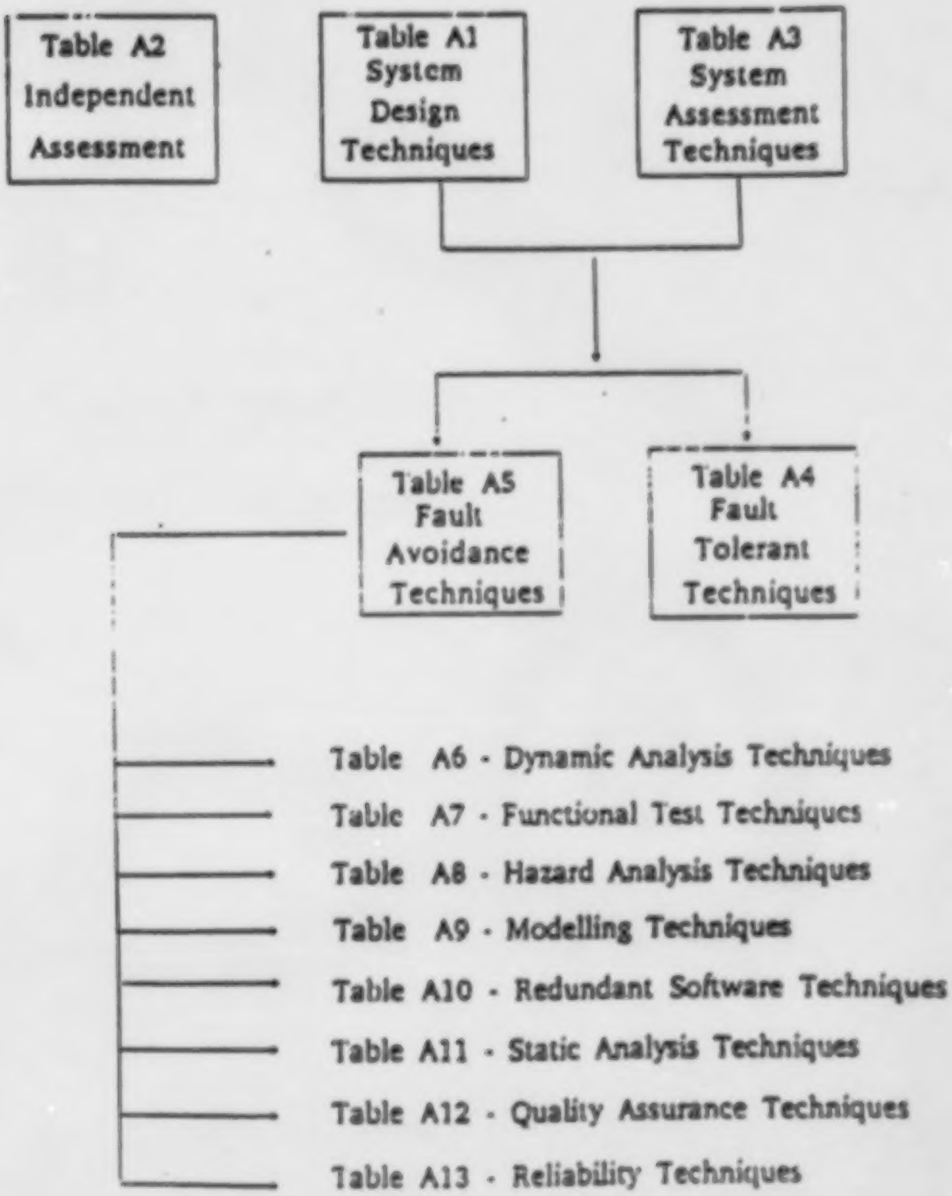


Table A2
Independent
Assessment
Table A1
System
Design
Techniques
Table A3
System
Assessment
Techniques
Table A5
Fault
Avoidance
Techniques
Table A4
Fault
Tolerant
Techniques
Table A6 - Dynamic Analysis Techniques
Table A7 - Functional Test Techniques
Table A8 - Hazard Analysis Techniques
Table A9 - Modelling Techniques
Table A10 - Redundant Software Techniques
Table A11 - Static Analysis Techniques
Table A12 - Quality Assurance Techniques
Table A13 - Reliability Techniques

The techniques selected from these tables should be detailed, along with the rationale behind the decision in the Quality Assurance Plan.

If tools exist for the techniques chosen and these tools have either been validated or proven by use then it is recommended that these automated tools be used

Each individual item in the table has a recommendation for each safety integrity level, 1 to 5. These recommendations can be:

'HR' - This technique is Highly Recommended for this safety integrity level. If this technique is not used then the rationale behind not using it should be detailed in the Quality Assurance Plan and authorised by the Authority Manager and agreed with the Assessor.

'R' - This technique is Recommended for this safety integrity level only when there is no 'HR' recommendation or where it forms part of a package of techniques.

'-' - This technique has no recommendation for or against being used.

'NR' - This technique is positively Not Recommended for this safety integrity level. If this technique is used then the rationale behind using it should be detailed in the Quality Assurance Plan and authorised by the Authority Manager and agreed with the Assessor.

A.4 Safety Integrity Levels and System Design Techniques

The table A1 below details the areas which it is recommended that the system design addresses for each safety integrity level.

Table A1 - System Design Techniques

Technique	Next Table	Safety Integrity Level				
		5	4	3	2	1
Fault Avoidance	A4	R	R	HR	HR	HR
Fault Tolerance	A5	R	R	R	R	R

A.5 Safety Integrity Levels and Assessment

Assessment by a third party is necessary for any software in safety integrity levels 1-4 according to Clause 4.3. Such third party assessment involves varying degrees of independence

The table A2 below details the degree of independence that is recommended for the assessment of software at each safety integrity level.

Table A2 - Independent Assessment Requirements

Technique/Method	Safety Integrity Level				
	5	4	3	2	1
Independent Third Party (Company)		R	HR	HR	HR
Independent Third Party (Department)	R	R	R	HR	HR
Independent Third Party (Personnel)	R	R	R		

Table A2 addresses the independence required of the verification, validation and assessment personnel with respect to those personnel involved in specification, design, development and testing.

Company - The personnel involved shall not belong to the same company.

Department - The personnel involved shall not belong to the same department.

Personnel - The personnel involved shall not be the same.

A.6 Safety Integrity Levels and Assessment Methods

The table A3 below details the areas which it is recommended that the assessment addresses for each safety integrity level.

Table A3 - Safety Properties to be Addressed

Technique	Next Table	Safety Integrity Level				
		5	4	3	2	1
Fault Avoidance	A4	HR	HR	HR	HR	HR
Fault Tolerance	A5	R	R	R	HR	HR

Each safety property in Table A3 is related to a table of relevant techniques. These relevant techniques are directed towards the design and development of software, they can however be used for the purposes for assessment.

Assessment is used to establish the integrity of the product and is a superset of validation. The software is therefore assessed against a wide variety of safety issues with respect to the system.

A.7 Fault Tolerant Methods

Table A4 - Fault Tolerant Methods

Technique	Forward Ref.	Safety Integrity Level 5	4	3	2	1
Defensive Programming	B.2.1				R	R
Error Correcting	B.2.2				NR	NR
Failure Testing	B.2.3		R	R	R	R
Fault Detection and Diagnosis	B.2.4			R	R	R
Return to Manual Operation	B.2.5			R	R	R
Re-Try Fault Recovery	B.2.6				R	R
~~Active Redundancy~~	~~B.2.7~~			~~R~~	~~R~~	~~R~~
~~Hot Stand By (Passive Redundancy)~~	~~B.2.8~~		~~R~~	~~R~~	~~R~~	~~NR~~
~~Cold Stand By~~	~~B.2.9~~		~~R~~	~~R~~		
Safety Bag	B.2.10			R	R	R
2 Version Programming	B.2.11		R	R	NR	NR
N Version Programming	B.2.12		R	R	NR	NR
Recovery Block	B.2.13		R	R	R	R
Backward Recovery	B.2.14		R	R	R	R
Forward Recovery	B.2.15		R	R	R	R

Table A4 - Fault Tolerant Methods (Contd)

Technique	Forward Ref.	Safety Integrity Level 5	4	3	2	1
Concurrent Simulation	B.2.16			R	R	R
Graceful Degradation	B.2.17	R	R	R	HR	HR
Knowledge Based Supervision	B.2.18			R	R	HR
AI - Fault Correction	B.2.19			NR	NR	NR
Dynamic Reconfiguration	B.2.20			NR	NR	NR

A.8 Fault Avoidance Techniques

Table A5 - Fault Avoidance Techniques

Techniques	Next Table	Safety Integrity Level 5	4	3	2	1
Dynamic Analysis	A6	R	R	R	HR	HR
Functional Test	A7	R	R	HR	HR	HR
Hazard Analysis	A8		R	HR	HR	HR
Modelling	A9	R	R	HR	HR	HR
Redundant Software	A10	R	R	R	HR	HR
Static Analysis	A11	R	R	R	HR	HR
Quality Assurance	A12	R	R	HR	HR	HR
Reliability	A13		R	R	HR	HR

A.8.1 Dynamic Analysis

Table A6 - Dynamic Analysis Techniques

Technique	Forward Ref	Safety Integrity Level				
		5	4	3	2	1
Boundary Value Analysis	B.3.1.1	R	R	HR	HR	HR
Error Guessing	B.3.1.2	R	HR	HR	HR	HR
Error Seeding	B.3.1.3			R	R	R
Performance Modelling	B.3.1.4		R	R	R	HR
Simulation	B.3.1.5	R	R	R	R	R
Test Coverage	B.3.1.6	R	R	R	HR	HR

A.8.2 Functional Test

Table A7 - Functional Test Techniques

Technique	Forward Ref	Safety Integrity Level				
		5	4	3	2	1
Cause Effect Diagrams	B.3.2.1		R	R	R	R
Prototyping/ Animation	B.3.2.2		R	R	R	R
Functional Testing	B.3.2.3	R	R	HR	HR	HR
Simulation	B.3.2.4	R	R	R	R	R

A.8.3 Hazard Analysis

Table A8 - Hazard Analysis Techniques

Technique	Forward Ref	Safety Integrity Level				
		5	4	3	2	1
Cause Effect Diagrams	B.3.3.1				R	R
Checklists	B.3.3.2		R	R	R	R
Event Tree Analysis	B.3.3.3		R	R	R	R
Failure Modes, Effects and Criticality Analysis	B.3.3.4		R	R	R	R
Fault Tree Analysis	B.3.3.5		R	R	R	R
Hazard and Operability Study (HAZOP)	B.3.3.6			R	R	R

Note : Preliminary hazard analysis should have already taken place in order to categorise the software into the most appropriate safety integrity level.

A.8.4 Modelling

Table A9 - Modelling Techniques

Technique	Ref.	Safety Integrity Level				
		5	4	3	2	1
Data Flow Diagrams	B.3.4.1	R	R	R	R	R
Finite State Machines / State Transition Diagrams	B.3.4.2			HR	HR	HR
Formal Mathematical Modelling	B.3.4.3			R	R	R
GO Charts	B.3.4.4			R	R	R
Performance Modelling	B.3.4.5		R	R	R	HR
Time Petri Nets	B.3.4.6			HR	HR	HR
Prototyping/ Animation	B.3.4.7	R	R	R	R	R
Reliability Block Diagram	B.3.4.8		R	R	R	R
Structure Diagrams	B.3.4.9	R	R	R	R	HR

Note : In risk levels 1, 2 and 3 one of the techniques used shall be either Formal Mathematical Modelling or Petri Nets.

A.8.5 Redundant Software

Table A10 - Redundant Software Techniques

Technique	Forward Ref	Safety Integrity Level 5	4	3	2	1
Checklists	B.3.5.1	R	R	R	R	R
Common Cause Failure Analysis	B.3.5.2			R	HR	HR
Markov Models	B.3.5.3		R	R	R	HR
Reliability Block Diagram	B.3.5.4		R	R	R	R

A.8.6 Static Analysis

Table A11 - Static Analysis Techniques

Technique	Forward Ref	Safety Integrity Level 5	4	3	2	1
Boundary Value Analysis	B.3.6.1	R	R	R	HR	HR
Checklists	B.3.6.2	R	R	R	R	R
Error Guessing	B.3.6.3	R	R	HR	HR	HR
Fagan Inspections	B.3.6.4			R	R	R
Formal Design Reviews	B.3.6.5	R	R	R	R	R
Formal Mathematical Modelling	B.3.6.6			R	R	HR
Formal Proof of Program	B.3.6.7			R	R	HR
Sneak Circuit Analysis	B.3.6.8			R	R	R
Walkthroughs	B.3.6.9		R	R	R	R

A.8.7 Quality Assurance

Table A12 - Quality Assurance Techniques

Technique	Forward Ref	Safety Integrity Level				
		5	4	3	2	1
Checklists	B.3.7.1	R	R	R	R	R
Fagan Inspections	B.3.7.2			R	R	R
Formal Design Reviews	B.3.7.3	R	R	R	HR	HR

A.8.8 Reliability

Table A13 - Reliability Techniques

Technique	Forward Ref	Safety Integrity Level				
		5	4	3	2	1
Failure Modes, Effects and Criticality Analysis	B.3.8.1		R	R	HR	HR
GO Charts	B.3.8.2			R	R	R
Reliability Block Diagrams	B.3.8.3		R	R	R	R
Reliability Growth Models	B.3.8.4		R	R	R	HR

[INFORMATIVE]

ANNEX B

Description of Methods for Design and Assessment

B.1 Introduction

This Annex describes all the techniques and methods introduced in Annex A. The list of techniques here and in Annex A, are not exhaustive. These two Annexes (Annex A and B) are intended to evolve with technology.

Some techniques are applicable in more than one instance and in such cases, the technique is detailed in the first occurrence of that technique.

Most of the techniques are described using a similar format, each having the following sections unless either they are not applicable or nothing exists (or is known) for that section.

Aim	- What is the aim of the technique
Description	- A short description of the technique.
Conditions	- Are there certain conditions which are required in order to apply the technique.
Major Advantages	- What are the major advantages gained by using this technique. for example, easy to apply, complete results).
Problems or Disadvantages	- Are there any special problems or difficulties connected to the use of this technique.
Results	- What is the form of the results and what does the technique really cover.
Relationship to Other Methods	- Does the technique overlap, make redundant or complement any of the other techniques.
References	- References to descriptions of the technique.

There also follows a detailed description of redundancy and diversity in Clauses B.1.1 and B.1.2.

B.1.1 Redundancy

Redundancy (multiple application of identically designed components) increases the reliability of a system and is a way of detecting or tolerating physical faults.

There are many different ways to apply redundancy. The two most important kinds of redundancy are the physical replication of components or complete systems ('parallel redundancy') and redundancy in the time domain ('serial redundancy').

In addition, there exists the concept of 'passive redundancy', denoting the provision of a standby computer which serves as a spare when the active computer fails. However, it has to be noted that the provision of a spare does not itself improve safety, it serves mainly for increasing the availability of the system.

When parallel redundancy is applied the replicated components fulfill concurrently identical actions. The result(s) of the operation are checked ('voted'; against each other. Depending on the degree of replication, faults can either be detected or masked. Since operations are performed concurrently faults can be masked without additional recomputations ('dynamic fault masking').

To avoid deterioration of the fault masking capability as more redundant (diverse) components fail, adaptive voting may be implemented. This is a technique in which the number of units taken into account in the vote can vary during the life of the system (hybrid or self-purging redundancy).

If serial redundancy is applied operations are performed sequentially more than once using a single hardware system. After completion of the computations results are voted and faults either detected or masked. Because of the sequential computations the time behaviour of the system is not as good as when parallel redundancy is used. Also there is a higher probability of common failure modes (especially those relying on hardware and system software faults). Time dependent failure modes are detected.

Although redundancy can be applied to any system component the best results can be achieved by replicating complete computers. If the computers are duplicated, the result of the comparison is the information that a physical fault has occurred in one of the computers. Even if the faulty computer can be identified by other measures, the healthy computer cannot continue to control the system. This is due to the fact that the results of the healthy computer cannot be compared to the results of the other computer as long as this computer remains damaged. Therefore a second physical fault is undetectable.

If the computers are triplicated, a majority voting can be done, which identifies the computer where a hardware fault has occurred. In case of a physical fault of one computer, the other two computers can continue to control the system. Their output results can still be compared such that a second physical fault can be detected.

The following considerations should clarify further the inherent characteristics of reliability and safety:

The same principle applies if when considering more than three computers in a system: If all of N systems have to deliver the same result to produce a valid output, the safety of the system is increased, if there are more computers and the reaction on disagreement of results does not lead to an unsafe situation. The reliability is decreased in that case since just one faulty component leads to a shutdown.

On the other hand, when majority voting is applied ($M > N/2$ out of N) the reliability of the overall system increases, but there is a certain probability that a majority of faulty components overrule a minority of correct ones. The system becomes more unsafe with smaller M (M nearer to $N/2$) and larger N.

Hardware redundancy may be used without voting in standby modes. In that case the system has to be switched over by means of a (reliable) emergency switch. Necessarily the system has to have selfchecking features. These may be implemented in hardware or software and assume some sort of internal redundancy and/or diversity.

A simple measure of redundancy is the multiplicity of identical systems or relevant parts of the system.

B.1.2 Diversity

In order to improve the safety of a system, the number of design faults, which remain undetected in the system, must be decreased. The main idea of diversity is to design and implement a system more than once in different ways, feed the different versions of the system with the same data and compare the results. If any difference in the time domain or in the value domain is detected, a design fault must be present and can be anticipated.

Only such design faults which are present in all versions of the system cannot be detected. Such the safety of a system may be improved, although an absolutely safe system cannot be achieved by this method.

For methods of diversity see Annex A, which describes methods to achieve diversity without giving a value judgement. For every system the advantages and disadvantages have to be evaluated carefully.

Possibilities to obtain high design Diversity are:

- Use different design and test-teams.
- Use different programming languages and compilers.
- Use different development tools.
- For maximum safety also use different types of computers and operating systems.
- Use a diverse specification (e. g. what the system should do and what the system must not do), diverse process models (e.g. operational model and safety model).
- Use separate computers, I/O-channels, power supplies etc.

B.2 Fault Tolerant Methods

B.2.1 Defensive Programming

Aim

To produce programs which detect anomalous control flow, data flow or data values during their execution and react to these in a predetermined and acceptable manner.

Description

Many techniques can be used during programming to check for control or data anomalies. These can be applied systematically throughout the programming of a system to decrease the likelihood of erroneous data processing.

Two overlapping areas of defensive techniques can be identified. Intrinsic error-safe software is designed to accommodate its own design shortcomings. These shortcomings may be due to plain error of design or coding, or to erroneous requirements. Following is a list of some of the defensive programming techniques:

a) Variables should be range checked.

b) Where possible, values should be checked for plausibility.

c) Parameters to procedures should be type, dimension and range checked at procedure entry.

These first recommendations (a-c) above help to ensure that the numbers manipulated by the program are reasonable, both in terms of the program function and physical significance of the variables.

d) Read-only and read-write parameters should be separated and their access checked. Functions should treat all parameters as read-only. Literal constants should not be write accessible. This helps detect accidental overwriting or mistaken use of variables.

Error tolerant software is designed to 'expect' failures in its own environment or use outside nominal or expected conditions, and behave in a predefined manner. Techniques include the following:

e) Input variables and intermediate variables with physical significance should be checked for plausibility.

f) The effect of output variables should be checked, preferably by direct observation of associated system state changes.

g) The software should check its configuration. This could include both the existence and accessibility of expected hardware and also that the software itself is complete. This is particularly important for maintaining integrity after maintenance procedures.

Some of the defensive programming techniques such as control flow sequence checking, also cope with external failures.

Major Advantages

a) Coverage of hardware and software failures.
b) Fairly easy to use and relatively low cost.

Problems or Disadvantages

It is difficult to quantify or access the defensive coverage gained by using the techniques and it should not be relied upon solely. It may also increase significantly the size of the code, and may slow down the execution speed.

Results

Increased confidence in the function performed by the program.

Relationship to other methods

Defensive methods can use a variety of failure detection methods and may be used in conjunction with any failure handling strategy. With a fail-stop approach, detected anomalies can be designed to de-activate a watchdog timer.

References:

1. Dependability of Critical Computer Systems - Part 1.
   E.F. Redmill (ed) Elsevier Applied Science 1988.

2. Dependability of Critical Computer systems - Part 2.
   E.F. Redmill (ed) Elsevier Applied Science 1988.

3. Software Engineering Aspects of Real-time Programming Concepts.
   E. Schoitsch, Computer Physics Communications
   41 (1986) North Holland, Amsterdam.

B.2.2 Error Correcting Codes

Aim

To detect and correct errors in sensitive information.

Description

For an information of n bits, a coded block of k bits is generated which enables t errors to be detected and corrected. Different types of code include

- Hamming codes,
- cyclic codes,
- polynomial codes.

Conditions

Encoding and decoding facilities (implemented in either hardware or software).

Major Advantages

In transmission systems, error correction is immediate, avoiding delays associated with re-transmission. In memory systems, random transient and memory chip failures can be tolerated.

Problems or Disadvantages

Additional time is required for message transmission or for memory error checking and correcting. Check blocks may become very large. Check codes designed to detect errors rather than correct errors are more efficient. The error correcting code technique is usually applied to data transmission systems and computer memory storage systems where time delays are critical. In principle it could also be implemented in software to protect sensitive blocks of data, but it should only be applied to individual words rather than checking the consistency of the whole data block, again because the check blocks become prohibitively large.

Results

Reduction in the number of data errors, correction delays are minimised.

Relationship to Other Methods

Information encoding, error detecting codes.

References

1. The technology of error-correcting codes. - E.R. Berlekamp, Proc. of the IEEE, 68, 5, 1980.

2. A short course on error correcting codes. - N.J.A. Sloane, Springer-Verlag, Wien, 1975.

B.2.3 Failure Testing (Assertion Programming)

Aim

To detect residual software design faults during execution of a program, in order to prevent safety critical failures of the system and to continue operation for high reliability.

Description

The assertion programming method follows the idea of checking a pre-condition (before a sequence of statements is executed, the initial conditions are checked for validity) and a post-condition (results are checked after the execution of a sequence of statements). If either the pre-condition or the post-condition is not fulfilled, the processing stops with an error.

For example,

```
assert <pre-condition>;
   action 1;
         :
         :
   action x;
assert <post-condition>;
```

Conditions

- Applicable in fail-safe systems only.

Major Advantages

No parallel resources, no voting and synchronisation necessary.

Problems or Disadvantages

Completeness and correctness of the pre- and post-conditions is not easy to establish, i.e. a wrong result could pass the assertions.

Relationship to Other Methods

See models of design diversity (N-version programming, recovery block scheme, safety bag technique).

References

1. A Discipline of Programming. - E.W. Diijkstra,
   Prentice Hall, 1976.

2. The Science of Programming. - D. Gries,
   Springer-Verlag, 1981.

3. Software Development - a rigorous approach. - C.B. Jones,
   Prentice-Hall, 1980.

B.2.4 Fault Detection and Diagnosis

Aim

To detect faults in a system, which might lead to a failure, thus providing the basis for countermeasures in order to minimise the consequences of failures.

Description

Fault detection is the process of checking a system for erroneous states (which are caused, as explained before, by a fault within the (sub)system to be checked). The primary goal of fault detection is to inhibit the effect of wrong results. A system which delivers either correct results, or no results at all, is called "self checking".

Fault detection is based on the principles of redundancy (mainly to detect hardware faults) and diversity (software faults). Some sort of voting is needed to decide on the correctness of results. Special methods applicable are assertion programming, N-version programming and the safety bag technique and on hardware level by introducing sensors, control loops, error checking codes, etc

Fault detection may be achieved by checks in the value domain or in the time domain on different levels, especially on the physical (temperature, voltage etc.), logical (error detecting codes), functional (assertions) or external level (plausibility checks). The results of these checks may be stored and associated with the data affected to allow failure tracking.

Complex systems are composed of subsystems. The efficiency of fault detection, diagnosis and fault compensation depends on the complexity of the interactions among the subsystems, which influences the propagation of faults.

Fault diagnosis isolates the smallest subsystem that may be identified. Smaller subsystems allow a more detailed diagnosis of faults (identification of erroneous states).

Conditions

Some form of redundancy must be available.

Major Advantages

Prerequisite for achieving reliability and safety.

Problems or Disadvantages

Some form of redundancy must be provided in the system, thus increasing the development and production costs.

Relationship to Other Methods

Basic to, and in strong interaction with, all methods to achieve safety and/or reliability. It is fundamental for fault compensating systems.

B.2.5 Return to Manual Operation

Aim

To provide the operator or supervisor with the information and the means to perform the function of the failed automatic control system.

Description

It is applicable to continuous processes with human supervision and/or partial operation. The general procedure is to:

- alarm the operator
- transfer the failed system from automatic operation
- allow manual process control by the operator

The operator would perform the same or possibly reduced set of the functions performed by the automatic system. Manual operation should be described in the operating procedures and has be approved by the safety authorities. In general, manual operation can only be permitted for a given period of time before the whole system has to be shut down.

Conditions

- The functions must be safely manageable by the operator with the information available in the control room

- The operator must be trained to follow the manual procedure

- The operation must be authorised by the safety authorities

Major Advantages

Increases the system availability, permits less redundancy in the automated controls and provides a diverse means of control.

Problems or Disadvantages

Requires continuous operator availability and specific training. Operators can be dangerously overloaded if too many systems fail at once.

Results

Higher system availability or reduced automation system complexity.

B.2.6 Re-Try Fault Recovery

Aim

To attempt functional recovery from a detected fault condition by re-try mechanisms.

Description

In the event of a detected fault or error condition, attempts are made to recover the situation by re-executing the same code. Recovery by re-try can be as complete as a re-boot and a re-start procedure or a small re-scheduling and re-starting task, after a software time-out or a task watchdog action. Re-try techniques are commonly used in communication fault or error recovery and re-try conditions could be flagged from a communication protocol error (check sum etc.) or from a communication acknowledgement response time-out.

Conditions

The time-criticality of the re-tried software must be greater than the time required for a re-try attempt to succeed. Re-try may result in the loss of real-time data and therefore full re-initialization may be needed. Re-trys should be limited in number.

Major Advantages

Usually relatively inexpensive to implement. Under normal, correct, operating conditions minimal software overhead incurred.

Problems or Disadvantages

Most real-time systems cannot accommodate the loss of data and the time delay of automatic re-boot. Re-try Fault Recovery is mainly used in handling communication error recovery.

Results

Relatively straight-forward recovery technique.

Relationships to Other Methods

Normally uses either software time-out checks or watchdog timers to trigger re-try attempts.

Assessment

Should be used with care and always with full consideration of the effect on time-critical events, and the effect of lost data during re-boot.

Reference

1. The Theory and Practise of Reliable System Design.
   D.P. Siewiorek and R.S. Schwarz, Digital Press.

B.2.7 Active Redundancy

Aim

The aim of active redundancy is twofold:

- To detect, as early as possible, failures of a hardware subsystem by comparing its results to a redundant one;
- To mask failures of a hardware subsystem, through the continued provision of the specified service with the other, redundant subsystems.

Description

Fault detection, masking of wrong results and propagation of the correct result are achieved by voting between simultaneously active units.

To this end, M redundant units (that is more than absolutely necessary to perform the required task) perform the task in parallel. When comparing the results, any difference indicates a failure of one of the redundant units.

When M is greater than two, the faulty unit can be diagnosed, and the required service can further be provided by the intact units (determined by an N out of M vote).

Since a single failure of the voting mechanism may lead to the failure of the group of redundant subsystems, the voting should operate in redundancy.

Since exact concurrency is not possible in all cases, certain time dependent software faults may be detected.

Synchronisation is a basic requirement for active redundancy:

The processors can be synchronised by means of hardware without using software. This is done by comparing the behaviour of the computers in each clock cycle. This method is often called microsynchronisation.

Another method is to use software methods to synchronise the computers. The programs are divided into certain processing steps. After each processing step the computers exchange significant results from these processing steps which are then compared with each other by means of software.

Conditions

Can be applied at various levels of the hierarchical system decomposition, for smaller and larger subsystems. Suitable for non-stop systems.

Fast communication links or shared memory between the redundant elements are necessary.

Precautions should be taken in the hardware against common cause failures.

Major Advantages

Performance in the time domain is very good, since the active units are executing concurrently. Faults can be detected and masked, before they spread to other subsystems.

Problems or Disadvantages

The following points need special attention:

- When the voting mechanism is not also performed in redundancy, it may be a critical point of failure.
- Synchronisation of redundant elements.
- Inexact voting.
- Additional cost.
- Handling of peripheral devices.

Relationship to Other Methods

Active redundancy is fundamental to many fault tolerance methods.

B.2.8 Hot Stand-by (Passive Redundancy)

Aim

To continue operation in case of and recover from failures (mainly hardware) and provide high availability.

Description

Only one system is active in the functional sense. This system is connected with a stand-by system by some communication link, and a switch or watchdog. There is no voting performed. The active unit may be:

- self-checking (and therefore able to initiate the switch to the passive (standby) system),
- controlled by the standby system,
- or switched off on detection of a fault by a third system (remote diagnosis, manual).

The standby system has to continue after a short time delay, which implies the use of methods to keep the databases and states in the stand-by system up-to-date with the controlling one. The performance in the time domain may be more critical then with active redundancy, because of the time needed to detect the failure of the controlling system switch-over time to the stand-by

Detection of design faults with software would require diversity. However, because there is no exact concurrency in all cases, the system may recover from certain time dependent software faults. Some types of software faults such as endless loops and deadlocks may be detected by a watchdog.

Conditions

- Extremely low MDT (Mean Down Time) and/or very high availability is required.
- Fast communication links or shared memory between the redundant elements are necessary.
- Precautions in the hardware against common cause failures should be taken.

Major Advantages

- No voting mechanisms are necessary.
- Switch-over time to the hot stand-by system is much lower than in cold stand-by mode.

Problems or Disadvantages

Fault detection requires additional mechanisms, which are difficult to make as efficient as with active redundancy.

The following points need special attention:

- Synchronisation of redundant elements.
- Consistency of databases.
- Total loss of dynamic (process) data may cause an intolerably long restoration time.
- Additional cost.

Relationship to Other Methods

See other models of redundancy (active redundancy, cold stand-by).

B.2.9 Cold Stand-by (Passive Redundancy)

Aim

To continue operation in case of and recover from (mainly hardware) failures, provide high availability.

Description

Only one system is active in the functional sense. This system is connected with a passive system by some switch or watchdog. There is no voting performed. The active unit may be:

- self-checking (and therefore able to initiate the switch to the passive (standby) system),
- controlled by the standby system,
- or switched on detection of a fault by a third system (remote diagnosis, manual).

The performance in the time domain is worse than with other forms of redundancy, because of switch-over time plus restoration time.

Because there is no concurrency, the system may recover from certain time dependent software faults. Some types of software faults such as endless loops and deadlocks may be detected by a watchdog.

Major Advantages

- No voting mechanisms are necessary (otherwise a separate proof of safety would be required).
- When a switch-over takes place, the complete history of the process is reconstructed and brought into a state consistent with the environment.

Problems or Disadvantages

- Fault detection requires additional mechanisms, which are difficult to make as efficient as with active redundancy.
- The complete history of the process gets lost when a switch-over takes place. This technique can therefore only be applied to systems (or subsystems) where no history exists or the history can be reconstructed from outside the system (subsystem) or the history is not necessary.
- Service interruption at switch-over may be more serious than in hot stand-by systems or with active redundancy, because of additional restoration time.

Relationship to Other Methods

See other models of redundancy, (hot stand-by redundancy).

B.2.10 Safety Bag

Aim

To detect residual software design faults during execution of a program, in order to prevent safety critical failures of the system, and to continue operation for high reliability.

Description

The safety bag technique follows the idea that an independent computer ("safety bag"), developed using a diverse specification, checks all results which are calculated by the main computer. These checks ensure, that the system does not enter an unsafe state. In addition the safety bag continuously monitors the state of the system. If a potentially hazardous state occurs, the system has to be brought back into a safe state either by the safety bag or by the main computer.

The possibility for common design faults in the diverse software is greatly reduced, since the specification of the safety bag is diverse from the specification of the software to be checked. This is usually necessary, since in a reasonably complex system a complete test (proving the absence of any faults) is impossible.

Conditions

Applicable in fail-safe systems only.

Major Advantages

- High degree of fault detection (including specification faults).
- Can be used in real-time systems.
- Since two independent computers are used, an erroneous state in one computer cannot influence the safe check in the other computer.

Problems or Disadvantages

- Faults can, in general, only be detected, not tolerated (no fault masking). This means that both computers have to be operative.
- Increased development and maintenance cost compared to a single version implementation.
- Special attention must be paid to voting and synchronisation.

Relationship to Other Methods

See other models of design diversity (2-version programming, assertion programming).

References

1. Using AI Techniques to Improve Software Safety
Proc. IFAC SAFECOMP 86,
Sarlat, France, Oct 1986, Pergamon Press 1986.

B.2.11 2-Version Programming

Aim

To detect residual software design faults during execution of a program, in order to prevent safety critical failures of the system.

Description

The 2-version programming technique follows the idea that a given program is implemented twice in different ways, starting from the same specification. All input values are given to the 2 versions and the results produced by the 2 versions are compared. If the 2 results have identical values in the value- and time-domains, then the result is considered to be valid and given to the process.

This method does not eliminate residual software design faults, but it provides a measure to detect most of them before they can affect safety.

Safety can only be affected, if both versions contain residual software faults, which behave equally in both versions, that is the two residual software faults will affect safety under the same circumstances at the same time in the same way.

Conditions

May be applied only in fail-safe systems, since in the case of a discrepancy between the two versions the system has to be brought into the safe fail-back state.

Major Advantages

Residual design and coding faults can be detected at runtime. This is usually necessary, since in a reasonably complex system a complete test (proving the absence of any faults) is impossible.

Problems or Disadvantages

- Faults can, in general, only be detected, not tolerated (no fault masking). This means that both computers have to be operative.
- Since only a single specification is used, specification errors may remain undetected.
- Additional software for voting and synchronisation is necessary.
- Increased development and maintenance cost compared to a single version implementation.

Relationship to Other Methods

See other models of design diversity (recovery block scheme, assertion programming, safety bag technique).

B.2.12 N-Version Programming

Aim

To detect and mask residual software design faults during execution of a program, in order to prevent safety critical failures of the system, and to continue operation for high reliability.

Description

The N-version programming technique follows the idea that a given program is implemented N times in different ways, starting from the same specification. All input values are given to the N versions, and the results produced by the N versions are compared. If in this comparison an absolute majority of results is established, in the value- and time-domains, then the majority result is considered to be valid and given to the process.

This method does not eliminate residual software design faults, but it provides a way to detect and mask them before they can affect safety.

Safety can only be affected, if an absolute majority of versions contain residual software design faults, which behaves equally in these versions, i.e. all these residual software design faults try to affect safety under the same circumstances at the same time in the same way.

Conditions

Recommended for safety relevant fault compensating systems.

Major Advantages

Residual design and coding faults can be detected and masked at runtime without loss of service. This is usually necessary, since in a reasonably complex system a complete test (proving the absence of any faults) is impossible.

Problems or Disadvantages

- Since only a single specification is used, specification errors may remain undetected.
- Additional software for voting and synchronisation is necessary.
- Increased development and maintenance cost compared to a single version implementation.

Relationship to Other Methods

See other models of design diversity (recovery block scheme, assertion programming, safety bag technique).

Reference

1. Dependable Computing : From Concepts to Design Diversity,
A. Avizienis, J.C. Laprie, Proc IEEE, Vol 74, 5, May 1986.

B.2.13 Recovery Block Scheme

Aim

To detect and mask residual software design faults during execution of a program, in order to prevent safety critical failures of the system, and to continue operation for high reliability.

Description

The recovery block scheme follows the idea, that a given procedure within a program is implemented N times and that an acceptance test is implemented, which determines if a result is valid. Version 1 is always executed. Version N+1 is executed, if the acceptance test identifies the result from version N as invalid. This is done as long as versions are available. If the results of the last version are also invalid, then the processing stops with an error

The original input data has to be available for the consecutive execution of all the diverse module versions.

```
Programming model:
        ensure <acceptance Test>
          by <module version 1>
            else by <module version 2>
            else by <module version x>
            else error;
```

Conditions

- Applicable in transaction oriented systems.
- Only for soft real-time or non-real-time systems.

Major Advantages

No parallel resources, no voting and synchronisation necessary.

Problems or Disadvantages

- Hardly applicable in real-time systems.
- Completeness of the acceptance test is not easy to establish, i.e. a wrong result could pass the acceptance test.

Relationship to Other Methods

See other models of design diversity (N-version programming, assertion programming, safety bag technique).

Reference

1. System Structure for Software Fault Tolerance. - B. Randell, IEEE Trans Software Engineering, Vol SE-1, No 2, 1975.

B.2.14 Backward Recovery

Aim

To provide correct functional operation in the presence of one or more faults.

Description

If a fault has been detected, the system is reset to an earlier internal state, the consistency of which has been proven before. This method implies saving of the internal state frequently at so-called well defined checkpoints. This may be done globally (for the complete database) or incremental (changes only between checkpoints). Then the system has to compensate for the changes which have taken place in the meantime by using journalling (audit trail of actions), compensation (all effects of these changes are nullified) or external (manual) interaction.

Conditions

It must be possible to define checkpoints, which occur frequently enough, and to check their consistency. It must be possible to restore all input signals since the last checkpoint, and to contain possible errors.

Major Advantages

Natural in transaction oriented systems.

Problems or Disadvantages

- Bad real-time behaviour, because the occurrence and duration of the recovery actions is non-deterministic.
- Old data may have become inconsistent with the evolving environment.

Relationship to Other Methods

See Recovery Block Scheme and Forward Recovery.

B.2.15 Forward Recovery

Aim

To provide correct functional operation in the presence of one or more faults.

Description

If a fault has been detected, the current state of the system is manipulated to obtain a state, which will be consistent some time later. This concept is especially suited for real-time systems with a small database and fast rate of change of the internal state. It is assumed, that at least part of the system state may be imposed onto the environment, and only part of the system states are influenced (forced) by the environment.

Conditions

It must be possible to reconstruct parts of the internal state from the external world, the other part has to be imposed from the system to the external world.

Major Advantages

Acceptable real-time behaviour.

Problems or Disadvantages

- It is not always possible to reconstruct the internal state from the external world and to impose the other part onto the external world.

- No history is available.

Relationship to Other Methods

Booting, switch-over to cold standby (passive redundancy).

B.2.16 Concurrent Simulation

Aim

The aim is two-fold:

- Continuously supervise the correctness of the process model, which was used when the system was designed.

- Try to forecast possibly hazardous consequences, in particular of human interactions.

Description

A complex technical process is simulated concurrently in the background or by another computerised system, calculating the possibly hazardous consequences of human interactions, events, or faults, and observing all deviations from the normal course of events. This concurrent simulation process is able to prepare countermeasures, to issue warnings and initiate actions to avoid dangerous system states (on a highly application dependent level of course).

Conditions

It must be possible to develop a sufficiently detailed process model, which takes into account not only the normal operating states, but also exceptional events and/or states.

Major Advantages

- The possibly hazardous effects of exceptional events and/or erroneous human interaction can be identified in advance of any effects on the external world.

- The correctness of the process model is continuously observed.

- The designer is forced to develop a detailed mental model of the process and to formalise it. Thus design errors will be reduced.

Problems or Disadvantages

- Increases the development effort significantly. Completeness and correctness of the simulation model are difficult to prove.

- May reduce reliability.

D.2.17 Graceful Degradation

Aim

To continue to provide the more important or critical services despite failures in the system. Upon each failure of a part of the system, the least important or least critical system functions may fail, whereas the more important services shall be maintained.

Description

If a fault occurs, it is often neither possible to enter a safe fall-back state nor to make a recovery in time.

The design ensures that should there be insufficient resources to continue all system functions after part of the system has failed, then the higher priority services continue to be provided, whereas less important services are allowed to fail.

By reducing the functionality it is possible to return to a safe state for a certain time, or to continue only the safety related functions. Non-safety related activities may be reduced, if there are difficulties in the time domain or in consumption of one of the major resources.

Conditions

The structure of the system must be such that upon failure of some component, the other components can continue to provide the more important, in particular the safety-related, functions.

Major Advantages

In general, this approach is less expensive than full fault-tolerance, where all system functions must be provided after a fault in the system.

Problems or Disadvantages

Graceful degradation schemes do not provide highly reliable systems, since some functions may fail in case of a fault.

Relationship to Other Methods

A cheaper reconfiguration strategy than full redundancy.

References

1. Space Shuttle Software - C.T. Sheridan,
   Datamation, Vol 24, July 1978

2. The Evolution of Fault-Tolerant Computing,
   Vol 1 of Dependable Computing and Fault Tolerant Systems,
   Edited by A. Avizienis, H. Kopetz, and J.C. Laprie,
   Springer-Verlag, ISBN 3-211-81941-X, 1987.

3. Fault Tolerance, Principle and Practices,
   Vol 3 of [2] above, T. Anderson and P.A. Lee,
   Springer-Verlag, ISBN 3-211-82077-9, 1987.

B.2.18 Knowledge-base Supervision

Aim

To force the system into a safe fall-back state if the software executes a path which has not been tested.

Description

A fault is detected and handled by some safety directed actions by checking the actual executed cases (inputs, states and results) against a knowledge base of assured (qualified) cases.

This knowledge base may be a record of all test cases.

It may also include ranges and calculated dependencies.

This approach assumes a profound knowledge of the technical process, proper modularisation or, as an optimal case, a formal description of a process.

Conditions

Ample storage and time must be available to memorise all test cases, and to compare the actual system behaviour at run-time to these test cases.

Major Advantages

The software is continuously monitored to execute only those sequences of actions, which have been tested.

Problems or Disadvantages

- Bad real-time behaviour, ample memory resources required.
- In a reasonably complex system, a complete test is not possible. If the software performs a correct sequence of actions, which however has not been among the test cases, the system will be stopped, resulting in reduced reliability.

B.2.19 AI - Fault Correction

Aim

To be able to react on possible hazards in a very flexible way by introducing a mix (combination) of methods and process models and some kind of on-line safety and reliability analysis.

Description

In particular fault forecasting (calculating trends), fault correction, maintenance and supervisory actions may be supported by AI-based systems in a very efficient way in diverse channels of a system, since the rules might be derived directly from the specifications and checked against these. Certain common faults which are introduced into specifications by implicitly already having some design and implementation rules in mind may be avoided effectively by this approach, especially when applying a combination of models and methods in a functional or descriptive manner.

The methods are selected such that faults may be corrected and the effects of failures be minimised, in order to meet the desired safety and reliability

Conditions

It must be possible to implement a set of complementary process models and to do some on-line safety analysis on at least two independent systems.

Major Advantages

- Flexible response, continuous monitoring of system states and conditions, high degree of diversity.

- Very profound knowledge of the technical process is needed, and a careful safety and reliability analysis has to be done.

Problems or Disadvantages

Additional cost for development and maintenance, because of redundancy in hardware and diversity in software, in particular for introducing AI-methods, which implies a considerable consumption of resources.

Relationship to Other Methods

Safety and Reliability Analysis, Redundancy for fault detection and failure containment, Concurrent Simulation.

B.2.20 Dynamic Reconfiguration

Aim

To maintain system functionality despite an internal fault

Description

The logical architecture of the system has to be such that it can be mapped onto a subset of the available resources of the system. The architecture needs to be capable of detecting a failure in a physical resource and then remapping the logical architecture back onto the restricted resources left functioning. Although the concept is more traditionally restricted to recovery from failed hardware units, it is also applicable to failed software units if there is sufficient 'run-time redundancy' to allow a software re-try or if there is sufficient redundant data to make the individual and isolated failure a little import.

Although traditionally applied to hardware, this technique is being developed for application to software and, thus, the total system. It must be considered at the first system design stage.

Conditions

To be applicable, the concept must be included from the inception of the system. The specification and design must accommodate it, and redundant resources must be included.

Major Advantages

Much increased system availability at full functionality if the potential is fully exploited.

Problems or Disadvantages

Achieving the full potential of this technique is difficult and has proved to be error prone itself. The correctness of error recovery software is acknowledged as being hard to establish. There is also an increased system cost for both extra resources and extra software.

Results

System availability increased substantially.

Relationship to Other Methods

This is the strongest variety of reconfiguration, being essentially an automated enhancement to 'hot standby', 'warm re-start' and other similar techniques. It is also stronger than 'graceful degradation', since full functionality is maintained.

References

1. Critical Issues in the Design of a Reconfigureable Control Computer.
   M. Schmid, J. Lam, R. Naro and K. Weir.
   FTCS 14 June 1984 IEEE 1984.

2. Assigning Processes to Processors : A Fault-tolerant Approach.
   June 1984 G. Kar and C.N. Nikolaou. Watson Research Centre, Yorktown

B.3 Fault Avoidance

B.3.1 Dynamic Analysis

B.3.1.1 Boundary Value Analysis

Some software errors are related to parameter limits or boundaries. An example is an integer value erroneously used from 1 to 256 instead of 0 to 255. Boundary value analysis directs attention towards the boundaries between equivalence classes. In addition to the selection of a test case inside an equivalence class, test cases on and close to the class limits must be included

Boundary value considerations also apply to white-box testing based on the code listed. Trivial errors are often related to loop counters, matrix sizes, tables stacks etc.

For example, if an input value is specified from 0 to 255, the equivalence class partitioning gives test cases for values:

- less than zero
- between zero and 255
- greater than 255

and the boundary value analysis adds test cases for the values:

- 1,   0,   1
254, 255, 256

If an input voltage is specified to be in the range of -5V to +5V, then test cases for

-5.1 V, -5.0 V, -4.9 V,
 5.1 V, 5.0 V, 4.9 V

The use of the value zero, in a direct as well as in an indirect translation, is often error-prone and demands special attention:

- zero divisor

- blank Ascii characters

- empty stack or list element

- null matrix

- zero table entry

Normally the boundaries for input have an direct correspondence to the boundaries for the output range. Test cases should be written to force the output to its limited values. Consider also, if it is possible to specify a test case which causes output to exceed the specification boundary values.

If output is a sequence of data, for example a printed table, special attention should be paid to the first and the last elements and to lists containing none, 1 and 2 elements.

B.3.1.2 Error Guessing

Testing experience and intuition combined with knowledge and curiosity about the system under test may add some uncategorised test cases to the designed test case set.

Special values or combinations of values may be error-prone Some interesting test cases may be derived from the inspection check-lists. It may also be considered whether the system is robust enough. Can the buttons be pushed on the front-panel too fast or too often? What happens if two buttons are pushed simultaneously?

B.3.1.3 Error Seeding

One way to distinguish whether a set of test cases is adequate, is the error seeding technique. Some known error types are inserted in the program, and the program is executed with the test cases under test conditions. If only some of the seeded errors are found, the test case set is not adequate. The ratio of found seeded errors to the total number of seeded errors is an estimate of the ratio of found real errors to total number errors. This gives a possibility of estimating the number of remaining errors and thereby the remaining test effort.

$$\frac{\text{Found seeded errors}}{\text{Total number of seeded errors}} = \frac{\text{Found real errors}}{\text{Total number of real errors}}$$

The detection of all the seeded errors may indicate either that the test case set is adequate, or that the seeded errors were too easy to find. The limitations of the method are that in order, to obtain any usable results, the error types as well as the seeding positions must reflect the statistical distribution of real errors.

B.3.1.4 Performance Modelling

Aim

To ensure that the working capacity of the system is sufficient to meet the specified requirements.

Description

The requirements specification includes throughput and response requirements for specific functions, perhaps combined with constraints on the use of total system resources. The proposed system design is compared against the stated requirements by:

- defining a model of the system processes, and their interactions.
- identifying the use of resources by each process (e.g. processor time, communications bandwidth, storage devices etc).
- identifying the distribution of demands placed upon the system under average and worst-case conditions.
- computing the mean and worst-case throughput and response times for the individual system functions.

For a simple system, an analytic solution may be possible whilst for more complex systems, some form of simulation is required to obtain accurate results

Before detailed modelling, a simpler 'resource budget' check can be used which sums the resources requirements of all the processes. If the requirements exceed designed system capacity, the design is infeasible. Even if the design passes this check, performance modelling may show that excessive delays and response times occur due to resource starvation. To avoid this situation engineers often design systems to use some fraction (e.g. 50%) of the total resources so that the probability of resource starvation is reduced.

Conditions

An accurate model of the system functions and their interactions is required. The analysis requires an estimate of the resource usage and elapsed time of each system function. These estimates can be obtained in several ways: comparison with existing systems, prototyping and benchmarking of time-critical systems, queuing and simulation applied to subsystems.

Major Advantages

Early identification of infeasible requirements.

Problems or Disadvantages

It may be difficult to obtain accurate estimates of function times and resource usage. The use of simulation demands that the system is modelled in detail, and this detail recorded in some simulation language. This may be a considerable programming exercise with its own reliability problems.

Relationship to Other Methods

May be used as a form of prototyping. Good systems analysis and design is a prerequisite for performance modelling.

B.3.1.5 Simulation

Aim

To test the function of a software system, together with its interface to the outside world, without allowing it to modify the real world in any way.

Description

The creation of a system, for testing purposes only, which mimics the behaviour of the system to be controlled by the system under test.

The simulation may be software only or a combination of software and hardware. It must:

- provide all the inputs of the system under test which will exist when the system is installed,
- respond to outputs from the system in a way which faithfully represents the controlled plant,
- have provision for operator inputs to provide any perturbations with which the system under test is required to cope.

When software is being tested the simulation may be a simulation of the target hardware with its inputs and outputs.

Conditions

Detailed knowledge, both practical and theoretical, of the system being controlled, with if possible empirical data on its behaviour.

Major Advantages

- Provides a means of performing functional tests of the system's behaviour in the event of catastrophes which could not otherwise be tested.
- Can provide a means of achieving 'long term' tests in a short period.
- Can provide a means of investigating behaviour at critical points by retarding the timescale or single stepping.

Problems or Disadvantages

- Requires knowledge of the specific application.
- Difficult to achieve independently of the software being tested.
- May require considerable resource both to create the simulation and to operate it.
- Depends on a model of the controlled system which could be flawed and is itself safety related.

Relationship to Other Methods

A form of functional testing.

References

1. A Software Simulator - An Aid to Plant Commissioning
   S. Nunns. EWICS TC7.

2. Physical Fault Simulation.
   F. Morillon. EWICS Document number WP460.

B.3.1.6 Test Coverage

Aim

To execute a percentage of the paths in the program at least once. On safety related systems the test coverage goal should be 100 percent.

Description

Structural testing gives certain possibilities of measuring the coverage of a set of test cases. The goal is that all executable paths through the program shall be covered by a test. It is often necessary to group paths into classes and test one path from each class. Test coverage measurement programs are relatively simple to develop. Test coverage analysers exist in many variations and are widely used by software quality assurance offices.

The program is instrumented by hand or by a processing program with counters and flags in all different branches. When a branch is executed the corresponding flag is up-dated. After test execution with the test cases under test, the flag values are listed, accumulated and compared to the optimum value dependent on the actual criterion.

Conditions

It must be possible to find all paths through the program, and to check whether all the paths have been executed.

Major Advantages

Very effective for detecting logic errors and has a larger coverage than other methods.

Problems or Disadvantages

- Exhaustive path testing is in general almost impossible.
  Does not find missing paths.

The technique pinpoints clearly areas of the code which are not tested properly, but even if the coverage is 1% or 100%, some error types will escape detection:

- Missing check of particular conditions, for example check for zero divisor.
- Inappropriate path selection, for example due to incorrect condition expression.
- Incorrect or missing actions.

Results

Correct execution of all program paths covered.

Relationship to Other Methods

Should be complemented with boundary value analysis.

References

1. Reliability of the Path Analysis Testing Strategy.
   W. Howden, IEEE Trans Software Engineering, Vol SE 3, 1976.

B.3.2 Functional Test Techniques

B.3.2.1 Cause Effect Diagrams

Aim

To model, in a diagrammatical form, the sequence of events that can develop in a system as a consequence of combinations of basic events.

Description

It can be regarded as a combination of fault-tree and event-tree analysis. Starting from a critical event, a cause-consequence graph is traced backwards and forwards. In the backwards direction it is equivalent to a fault tree with the critical event as the given top event. In the forward direction the possible consequences arising from an event are identified. The graph can contain vertex symbols which describe the conditions for propagation along different branches from the vertex. Time delays can also be included. These conditions can also be described with fault trees. The lines of propagation can be combined with logical symbols, to make the diagram more compact. A set of standard symbols are defined for use in cause consequence diagrams. The diagrams can be used to compute the probability of occurrence of certain critical consequences.

Major Advantages

- The method is systematic.
- The method is comprehensive, and is capable of handling most aspects of the development.
- The diagram is easy to understand.
- It is possible to make probability computations on the basis of a cause consequence diagram.
- The method is computer supported.

Problems or Disadvantages:

- The diagram can be very large.
- Difficult to describe continuous development.
- The possibility of rejoining development lines may hide some peculiarities of separate development sequences.

Relationship to Other Methods

Similar to event trees and utilises fault tree analysis.

References

1. The Cause Consequence Diagram Method as a Basis for Quantitative Accident Analysis. - D.S. Nielsen, Riso-M-1374, 1971.

B.3.2.2 Prototyping / Animation

Aim

To check the feasibility of implementating the system against the given constraints. To communicate the specifier's interpretation of the system to the customer, in order to locate misunderstandings.

Description

A sub-set of system functions, constraints, and performance requirements are selected. A prototype is built using high level tools. At this stage, constraints such as the target computer, implementation language, program size, maintainability, reliability and availability need not be considered. The prototype is evaluated against the customer's criteria and the system requirements may be modified in the light of this evaluation.

Conditions

Existence of a system specification containing quantified objectives and an explicit definition of the system function. Adequate channels of communication between the customer and the supplier.

Major Advantages:

- Better communication with the customer.
- Early detection of problems (contradictions, omissions, impossibilities).
- Can check the feasibility of new ideas or techniques.

Problems or Disadvantages:

- Unnecessary and expensive if the problem is well understood.
- Needs tools for quick implementation.
- There is a danger that the prototype may be used in the final product.

Results

Better agreement between the customer and supplier on the system functionality. More confidence that the system is feasible.

Relationship to other methods

Formal specification methods, performance simulation. May be tested in conjunction with a simulation of the plant connected to the system.

References:

1. Proc. Working Conference on Prototyping, Namur Oct 1983, Budde et al, Springer-verlag, 1984.

2. Using an executable specification language for an information system. - S. Urban et. al, IEEE Trans Software Engineering, Vol. SE-11 No. 7 July 1985.

B.3.2.3 Functional Testing

Functional testing or black-box testing is testing based on a specification of all possible inputs to and outputs from the software under test as well as description of the processing of the data. An example of pure functional testing is the functional part of the acceptance test, where the end-user tries out the features of the overall system.

Test. based on functional specifications are used at different levels. The software under test can be the whole system or part of it (a module or a subroutine). For functional acceptance test the input and output conditions are specified in the system requirement specification. Test case design for module testing uses the module specification as basis, either a detailed design description or the module heading available in the code listing. The principles for functional test case design are:

equivalence class partitioning

boundary value analysis

cause-effect graphing

B.3.2.4 Simulation

See Clause B.3.1.5 of this Annex.

B.3.3 Hazard Analysis

B.3.3.1 Cause Effect Diagrams

See Clause B.3.2.1 of this Annex.

B.3.3.2 Checklists

The purpose of checklists is to provide a stimulus to critical appraisal of all aspects of the system rather than to lay down specific requirements. To this end, many of the questions are of a general nature and the Assessor must interpret them as seems most appropriate to the particular system being assessed.

To accommodate wide variations in software and systems being validated, most checklists contain questions which are applicable to many types of system. As a result there may well be questions in the checklist being used which are not relevant to the system being dealt with and which should be ignored. Equally there may be a need, for a particular system, to supplement the standard checklist with questions specifically directed at the system being dealt with.

In any case it should be clear that the use of checklists depends critically on the expertise and judgement of the engineer selecting and applying the checklist. As a result the decisions taken by the engineer, with regard to the checklist(s) selected, and any additional or superfluous questions should be fully documented and justified. The objective is to ensure that the application of the checklists can be reviewed and that the same results will be achieved unless different criteria are used.

The object in completing a checklist is to be as concise as possible. When extensive justification is necessary this should be done by reference to additional documentation. Pass, Fail and Inconclusive, or some similar restricted set of tokens should be used to record the results for each question. This conciseness greatly simplifies the process of reaching an overall conclusion as to the results of the checklist assessment.

References for Checklists

1. Programmable Electronic Systems in Safety Related Applications, Health and Safety Executive, Her Majesty's Stationary Office, London 1987.

2. Guidelines for the Assessment of the Safety and Reliability of High Integrity Industrial Computer Systems, EWICS TC7, WP 489, 6th October 1987.

3. The Art of Software Testing, G.J. Myers, Wiley, 1987.

4. The Complete Guide to Software Testing, W. Hetzel, Collins, 1984.

5. Software for Computers in the Safety Systems of Nuclear Power Stations, IEC 880, 1986.

B.3.3.3 Event Tree Analysis

Aim

To model, in a diagrammatical form, the sequence of events that can develop in a system after an initiating event, and thereby indicate how serious consequences can occur.

Description

On the top of the diagram is written the sequence conditions that are relevant in the development following the initiating event which is the target of the analysis. Starting under the initiating event, one draws a line to the first condition in the sequence. There the diagram branches off into a 'yes' and a 'no' branch, describing how the future developments depend on the condition. For each of these branches, one continues to the next condition in a similar way. Not all conditions are, however, relevant for all branches. One continues to the end of the sequence, and each branch of the tree constructed in this way represents a possible consequence. The event tree can be used to compute the probability of the various consequences based on the probability and number of conditions in the sequence.

Conditions

A knowledge of the potential impact of events on the system.

Major Advantages:

- The tree is easy to draw when the sequence of events is established.
- The tree is easy to understand.
- It is easy to make probability computions on the basis of an event tree.

Problems or Disadvantages

- It may be difficult to identify the complete sequence of conditions.
- It is difficult to take into account different failure modes.
- It is difficult to take into account dependent failures, common equipment, and common fault interaction.
- The event tree can become very large.

Relationship to Other Methods

Similar to cause-consequence (cause-effect) diagrams. Complementary to fault tree analysis.

References:

1. Event Trees and their Treatment on PC Computers,
N. Limnious and J.P. Jeannette,
Reliability Engineering, Vol. 18, No. 3 1987.

B.3.3.4 Failure Modes, Effects and Criticality Analysis

Aim

To rank the criticality of components which could result in injury, damage or system degradation through single-point failures in order to identify those components which might need special attention and control measures during design or operation.

Description

Criticality can be ranked in many ways. The most laborious method is described by the Society for Automotive Engineers (SAE) in ARP 926. In this procedure, the criticality number for any component is indicated by the number of failures of a specific type expected during each million operations occurring in a critical mode. The criticality number is a function of nine parameters, most of these have to be measured. A very simple method for criticality determination is to multiply the probability of component failure by the damage that could be generated: this method is similar to simple risk factor assessment.

Conditions

A detailed system description of functions, conditions and (hardware) components is needed.

Major Advantages

Can be used for evaluation of system component failure which may result in a potential loss of life, loss of system functioning or availability, or excessive unscheduled maintenance.

Problems or Disadvantages

The detailed criticality ranking method as described by the SAE requires a great deal of effort. In simplified methods, the essential criticality attribute "potential damage" may be missing. For new, unique systems, component failure data will be less than adequate for a quantitative criticality ranking. A specific component may have several failure modes with only certain ones possibly causing safety problems or harm.

Results:

- helps to identify components which require more intensive analysis regarding hazard control or fail-safe design
- helps to identify components which need tight quality control, protective handling, etc.
- helps to assess special requirements to be included in the specification for suppliers.
- helps to establish measurable quality attributes and acceptable criteria for critical components
- helps to identify where hazard control resources could be applied most effectively

Relationship to Other Methods

FMECA is an extension of FMEA. Logic Trees, network analysis and (Preliminary) Hazard Analysis can also be used to determine items which may be critical.

B.3.3.5 Fault Tree Analysis

Aim

To aid in the analysis of events, or combinations of events, that will lead to a hazard or serious consequence.

Description

Starting at an event which would be the immediate causes of a hazard or serious consequence (the 'top event') analysis is carried out along a tree path. Combinations of causes are described with logical operators (and, or, etc). Intermediate causes are analysed in the same way, and so on back to basic events where analysis stops.

The method is graphical, and a set of standardised symbols are used to draw the fault tree. It is mainly intended for the analysis of hardware systems, but there have also been attempts to apply this approach to software failure analysis.

Conditions

Causes and effects of events must be known or calculable.

Major Advantages:

- The method is systematic.
  It is well suited as a basis for the computation of the probability of the top event, if one knows the probabilities of the basic events. There are a number of computer application packages which provide this computation.
- There is available computer support to construct fault trees.
- There is wide experience in the use of this method in safety analysis.

Problems or Disadvantages:

- The fault tree may get very large and complex.
- Dynamic aspects and time aspects are not addressed particularly well.
- The method concentrates its attention to specific top events, and is therefore not well suited to revealing other serious consequence.

Relationship to Other Methods

Uses failure modes and effects analysis (B.3.3.4).

References:

1. System Reliability Engineering Methodology: A Discussion of the State of the Art. - J.B. Fussel and J.S. Arend, Nuclear Safety 20(5), 1979.

2. Fault Tree Handbook. W.E. Vesely et. al., NUREG-0942,
Division of System Safety Office of Nuclear Reactor Regulation,
U.S. Nuclear Regulatory Commission Washington, D.C. 20555, 1981.

3. Reliability Technology. A.E. Greene and A.J. Bourne,
Wiley-Interscience, 1972.

B.3.3.6 Hazard and Operability Study (HAZOP)

Aim

To establish, by a series of systematic examinations of the 'component' sections of the computer system and its operation, failure modes which lead to result in potentially hazardous situations and/or events to the system being controlled.

The hazardous events on the controlled systems being fire, explosion, release of toxic material (chemical or nuclear) or serious economic loss.

It is assumed that the hazardous events on the system being controlled have been identified in a separate Hazard Analysis and the consequence of the hazardous events classified into degrees of seriousness.

The analysis which covers all stages of the project life-cycle, from specification through design to maintenance and modification, and is intended to identify at each stage events/failure modes which could lead to potential hazards and thus eliminate them.

Description

The analysis is carried out by a team of engineers (covering computer, instrument, electrical, process, safety and operational disciplines) led by a trained specialist in hazard analysis techniques in a series of scheduled meetings

It is important that a fixed time schedule is allocated within the project for the meetings; each one scheduled for at least half a day and for effectiveness no more than four per week to allow for maintaining the flow of accompanying documentation.

Prior to the study, agreed checklists for the systematic examination will have been compiled and for each section of the system leading questions such as; what if it happens? how can it happen? when can it happen? does it matter? are asked. When positive answers are given further questions are asked, such as; what can be done about it? when must it be done? is there an alternative? etc.

For the first part of the analysis it is suggested that the computer installation as a whole is examined. The second and subsequent parts involve detailed examination of the relevant parts of the computer systems itself.

At each part or stage of the analysis, the objective is to identify failure modes which lead to potential hazards on the controlled system and the degree of their effect. The major component parts of the computer system are further sub-divided and where necessary subjected to a separate analysis.

At suitable points throughout the systematic analysis, action review meetings will be arranged.

It is essential that comprehensive records of the meetings are kept, for they will form a substantial part of the system hazard/safety dossier.

After a number of meetings it is suggested that a review meeting be held to ensure that actions are followed up, modifications suggested during the Study meetings are incorporated into the study etc.

Conditions

- This technique needs a trained specialist in hazard analysis techniques to lead the team of engineers who have the knowledge of the 'system' being controlled and the computer system being used.

- Time is allocated to carry out the meetings on a regular basis.

- Formal reviews are carried out throughout the project.

Major Advantages

- It identifies potential hazards at every stage of the project life-cycle and their effects.

- It allows the effect of failure to be eliminated or reduced to an acceptable level at every stage of the project.

- It highlights the need for any special procedures or corrections to existing ones.

- It may help the design team to decide on the need for or increase of redundancy, diversity.

- It may contribute to the software specification, highlighting the need for special routines, modifications, or deletion of routines etc.

- It focuses on the sensitive areas of the system which, on failure could lead to potentially hazardous consequences and this highlights where effort can be most effectively applied to improve the system safety, reliability and availability.

- It provides a series of documents which will form part of the system safety dossier.

Problems or Disadvantages

- It takes time to carry out the study and with the expertise required will increase the cost of the project.

- It needs a rigid scheduling of meetings to be cost effective - possibly at times within the project when this is not easy to achieve.

- It needs someone of sufficient technical ability to maintain the documentation flow.

Results

- Identification of failure modes.
- Classification of effect of potential hazards.
- Concise documentation.
- Agreed actions and modification.
- Information for further analysis and quantification..

Relationship to other methods

Will provide information for input to FMEA, FTA etc.

References:

1. Reliability and Hazard Criteria for Programmable Electronic Systems in the Chemical Industry. - E. Johnson. Proc of 'Safety and Reliability of PES'. PES 3 Safety Symposium. B.K. Daniels (ed). 28-30 May 1986. Guernsey Channel Islands. Elsevier Applied Science 1986.

2. Reliability Engineering and Risk Assessment. E.J. Henley and H. Kumamoto. Prentice Hall 1981.

3. Systems Reliability and Risk Analysis. E.G. Frenkel. Martinus Nijhogg 1984.

4. Software Safety. - N.G. Leveson. 'Resilient Computer Systems'. T.Anderson (ed). Collins 1985 .

5. Software Safety in Computer Controlled Systems. N.G. Leveson. Computer. Feb. 1984.

6. Risk Assessment of Computer Controlled Systems. - M.O. Fryer. IEEE Trans Software Engineering. Vol SE-11. Jan 1985.

B.3.4 Modelling

B.3.4.1 Data Flow Diagrams

Data Flow Diagrams document how data input is transformed to output, with each stage in the diagram representing a distinct transformation.

Data flow diagrams are made up of three components.

1 Annotated Arrows	- represent data flow in and out of the transformation centres with the annotations specifying what the data is.
2. Annotated Bubbles	- represent transformation centres with the annotation specifying the transformation.
3. Operators (AND, XOR)	- These operators are used to link the annotated arrows.

Data flow diagrams describe how an input is transformed to an output. They do not, and should not, include control information or sequencing information. Each bubble can be considered as a stand alone black box which, as soon as its inputs are available, transforms them to its outputs.

One of the principle advantages of data flow diagrams is that they show transformations without making any assumptions about how these transformations are implemented.

The preparation of data flow diagrams is best approached by considering system inputs and working towards system outputs. Each bubble must represent a distinct transformation - its output should, in some way, be different from its input. There are no rules for determining the overall structure of the diagram and constructing a data flow diagram is one of the creative aspects of system design. Like all design, it is an iterative process with early attempts refined in stages to produce the final diagram.

References

1. Software Engineering. I. Sommerville, Addison-Wesley 1982.
ISBN 0-201-13795-X.

B.3.4.2 Finite State Machines / State Transition Diagrams

Aim

To define or implement the control structure of a system.

Description

Many systems can be defined in terms of their states, their inputs, and their actions. Thus when in state S1, on receiving input I a system might carry out action A and move to state S2. By defining a system's actions for every input in every state we can define a system completely. The resulting model of the system is called a Finite State Machine. It is often drawn as a so-called state transition diagram showing how the system moves from one state to another, or as a matrix in which the dimensions are state and input and the matrix cells contain the action and new state resulting from receipt of the input in the given state.

Where a system is complicated or has a natural structure this can be reflected in a layered FSM.

A specification or design expressed as an FSM can be checked for completeness (the system must have an action and new state for every input in every state), for consistency (only one state change is defined for each state/input pair) and reachability (whether or not it is possible to get from one state to another by any sequence of inputs). These are important properties for critical systems and they can be checked. Tools to support these checks are easily written. Algorithms also exist that allow the automatic generation of test cases for verifying an FSM implementation or for animating an FSM model.

Conditions

Use of FSMs requires that the system being developed can be modelled as a black box taking as input a sequence of stimuli. This is common of many process control systems and can in certain circumstances be applied to systems taking asynchronous inputs from different sources.

Major Advantages

FSMs allow certain important properties to be checked mechanically and reliably. They are also simple to work with.

Results

The use of FSMs results in a model (design or specification) of a system in a semi-formal form which can then be animated. Such a model can then be easily implemented in software and the software tested with test cases that can be generated automatically.

Relationship to Other Methods

A simple yet powerful technique that should be considered for event driven systems.

References

1. Introduction to the theory of finite state machines, A. Gill, McGraw-Hill 1962.

B.3.4.3 Formal Mathematical Modelling

Aim

The development of software in a way that is based on mathematics. This includes formal design and formal coding techniques.

Description

Formal Mathematical Modelling provides a means of developing a description of a system at some stage in its development specification, design or code. The resulting description takes a mathematical form and can be subjected to mathematical analysis to detect various classes of inconsistency or incorrectness. Moreover, the description can in some cases be analysed by machine with a rigour similar to the syntax checking of a source program by a compiler, or animated to display various aspects of the behaviour of the system described. Animation can give extra confidence that the system meets the real requirement as well as the formally specified requirement.

A formal method will generally offer a notation (generally some form of discrete mathematics being used), a technique for deriving a description in that notation, and various forms of analysis for checking a description for different correctness properties.

Conditions

Formal techniques generally require well trained personnel. A pre-requisite for training is some fluency (or additional training) in discrete mathematics, typically set theory and predicate calculus.

Major Advantages

Formal techniques increase the likelihood that their output (specification, design or code) is correct (see Clause B.3.6.7), that is internally consistent, is consistent with some prior model, possesses certain desirable properties, does not possess certain undesirable properties, and so on. This increased likelihood arises from the possibility of proving these properties mathematically from a mathematically expressed description.

Problems and Disadvantages

Formal techniques suffer from a number of disadvantages. Firstly, they can be more difficult and more time consuming to use; although these disadvantages may well be outweighed by the criticality of the application. Secondly, they are generally less well supported than, say, the systematic methods in terms of tools; this means that, as with the systematic methods, a great deal of work must be done manually (and hence with a risk of human error) rather than by a computer (and hence with greater confidence). The tool situation is however changing and for more 'popular' methods tool support is starting to become available.

B.3.4.4 GO Charts

Aim

Reliability analysis of complex systems (including components with two or more failure modes) mainly during the design stage.

Description

Seventeen fundamental operators are available to model the system components. Utilising the library of operators available a GO graph is constructed for each application. Each component of the system to be analysed is modelled by an operator. Each operator allows a particular class of components to be described by the following information:

- input signals to the component
- operational mode
- output signals from the component

Following the structure of the system, the inputs and outputs of different operators are connected, to construct the GO graph which mirrors the original structure. Finally, a probability is associated with the different states of the operators, and the graph is executed for reliability assessment.

Conditions

The system components must be statistically independent.

Major Advantages

Reduction in the time spent for the analysis of complex systems with multiple failure conditions. Easy to use, flexible and capable of modelling.

Problems and Disadvantages

The seventeen operators used by the GO methodology may not represent all the failure models of all components of a real system. In consequence, it can only be used for a preliminary analysis, and different reliability models are necessary to deal with more complex fault models. It is poor from a quantitative viewpoint.

Results

Qualitative analysis of complex systems with multi-state components.

Relationship to Other Methods

Should be complemented with a Fault Tree or Markov techniques for a quantitative reliability analysis.

References

1. An Introduction to GO. - B. Gateley,
Kaman Science Corporation, K-75, 94U (R), 1975.
1500 Garden of the Gods Road, Colorado Springs, Colorado 80907.

B.3.4.5 Performance Modelling

See Clause B.3.1.4 of this Annex.

B.3.4.6 Time Petri Nets

Aim

To model relevant aspects of the system behaviour and to assess and possibly improve safety and operational requirements through analysis and re-design.

Description

Petri nets belong to a class of graph theoretic models which are suitable for representing information and control flow in systems exhibiting concurrency and asynchronous behaviour.

A Petri net is a network of places and transitions. The places may be 'marked' or 'unmarked'. A transition is 'enabled' when all the input places to it are marked. When enabled, it is permitted (but not obliged) to 'fire'. If it fires, the input marks are removed, and each output place from the transition is marked instead.

The potential hazards an be represented as particular states (markings) in the model. Extended Petri nets allow timing features of the system to be modelled. Although 'classical' Petri nets concentrate on control flow aspects, several extensions have been proposed to incorporate data flow into the model.

Conditions

Mainly applicable to systems dominated by control aspects or concurrent asynchronous activities.

Major Advantages

Method is easy to understand and can provide thorough analysis for small systems. The model is (potentially) executable and this can be used to validate the adequacy of the model. Some classes of fault can be modelled together with their impact on safety (e.g. whether a hazardous state can be reached). Some (rather limited) guidance can be obtained on changes that can be made to the model to make the system tolerant to those faults.

Problems or Disadvantages

The properties of the whole system are far from obvious from the net's appearance (except very simple nets). The analytic methods work well on small networks or small parts of large networks, but is computationally intractable for large systems. Direct execution is of little help in demonstrating safety as only a small proportion of the overall set of states can be tested.

Results

A model of functional behaviour, an analysis of the safety of the specification, an analysis of the effect of some classes of fault, identification of those faults that threaten system safety.

Relationship to Other Methods

CSP and CCS (see Clauses D.3.1.4 and D.3.1.5) are alternative means of modelling and analysing system behaviour. Temporal logic can be used as alternative to demonstrate that the system specification (and design) is safe..

References

1. Net Theory and Applications. - W. Brauer (ed).
   Lecture Notes in Computer Science,
   Vol 84, Springer 1980.

2. Petri Net Theory and Modelling of Systems. - J.L. Peterson,
   Prentice Hall, 1981.

3. Safety Analysis using Petri Nets,
   N. Leveson and J. Stolzy,
   Proc. FTCS 15, Ann Arbor Michigan,
   June 1985, IEEE 1985.

4. A tool for requirements specification and analysis of Real time software
   based on timed petri nets,
   S. Bologna, F. Pisacane, C Ghezzi, D. Mandrioli,
   Proc. SAFECOMP 88, 9-11 Nov. 1988.
   Fulda Fed. Rep. of Germany 1988.

B.3.4.7 Prototyping / Animation

See Clause B.3.2.2 of this Annex.

B.3.4.8 Reliability Block Diagram

Aim

To model, in a diagrammatical form, the set of events that must take place and conditions which must be fulfilled for a successful operation of a system or a task.

Description

The target of the analysis is represented as a success path consisting of blocks, lines and logical junctions. A success path starts from one side of the diagram and continues via the blocks and junctions to the other side of the diagram A block represents a condition or an event, and the path can pass it if the condition is true or the event has taken place. If the path comes to a junction it continues if the logic of the junction is fulfilled. If it reaches a vertex, it may continue along all outgoing lines. If there exists at least one success path through the diagram the target of the analysis is operating correctly.

Conditions

Conditions and events must be analysed to allow structure of diagram to be determined.

Major Advantages

- The method is systematic
- The diagram is more compact than a fault tree
- A block diagram may follow the physical structure of the target system quite closely
- It is suited as a basis for the computation of the probability of the top event, if one knows the probabilities of the blocks. Computer programs are available which make this computation.
- It is easy to draw a fault tree on the basis of a block diagram

Problems or Disadvantages

- It addresses success rather than failures, the latter being more relevant to safety
- It is difficult to take into account different failure modes. This may lead to omission of important conditions.
- The method is not as widely used or as widely computer-supported as fault tree analysis.

References

1. System Reliability Engineering Methodology : A Division of the State of the Art. - J.B. Fussel and J.S. Arend, Nuclear Safety 20(5), 1979.

2. Fault Tree Handbook. W.E. Vesely et al. NUREG-0942, Division of System Safety Office at Nuclear Reactor Regulation, U.S. Nuclear Regulatory Commission, Washington, D.C. 20555 1981.

D.3.4.9 Structure Diagrams

Structure Diagrams are a notation which complements Data Flow Diagrams. They describe the programming system and a hierarchy of parts and display this graphically, as a tree. They document how elements of a data flow diagram can be implemented as a hierarchy of program units.

A structure chart shows relationships between program units without including any information about the order of activation of these units.

It is drawn using the following three symbols:

1. A rectangle annotated with the name of the unit.
2. An arrow connecting these rectangles.
3. A circled arrow, annotated with the name of data passed to and from elements in the structure chart. Normally, the circled arrow is drawn parallel to the arrow connecting the rectangles in the chart.

From any non trivial data flow diagram, it is possible to derive a number of different structure charts.

Structure charts derived from data flow diagrams represent a first level structure of the system, where each box on the structure chart represents a bubble in the data flow diagram. Naturally, deeper levels can be described using the same technique.

References

1. Software Engineering, I. Sommerville, Addison-Wesley 1982. ISBN 0-201-13795-X.

2. Structured Design, L.L Constantine and E. Yourdon, Englewood Cliffs, New Jersey, Prentice Hall, 1979.

3. Reliable Software Through Composite Design, G.J Myers, New York, Van Nostrand, 1975.

4. Techniques of Program Structure and Design, E. Yourdon, Englewood Cliffs, New Jersey, Prentice Hall, 1975.

B.3.5 Redundant Software

B.3.5.1 Checklists

See Clause B.3.3.2 of this Annex.

B.3.5.2 Common Cause Failure Analysis

Aim

To identify potential failures in redundant systems or redundant subsystems which would undermine the benefits of redundancy because of the appearance of the same failures in the redundant parts at the same time.

Description

Computer systems intended to take care of the safety of a plant often use redundancy in hardware and majority voting. This technique is used to avoid random component failures, which would tend to prevent the correct processing of data in a computer system.

However, some failures can be common to more than one component. For example if a computer system is installed in one single room, shortcomings in the air-conditioning, might reduce the benefits of redundancy. The same is true for other external effects on the computer system such as: fire, flooding, electromagnetic interference, plane crashes, and earthquakes. The computer system may also be affected by incidents related to operation and maintenance. It is essential, therefore, that adequate and well documented procedures are provided for operation and maintenance. Extensive training of operating and maintenance personnel is also essential.

Internal effects are also major contributors to Common-Cause Failures (CCF). They can stem from design errors in common or identical components and their interfaces, as well as aging of components. CCF-Analysis has to search the system for such potential common failures. Methods of CCF-Analysis are general quality control, design reviews, verification and testing by an independent team, and analysis of real incidents with feedback of experience from similar systems. The scope of the analysis, however, goes beyond hardware. Even if 'diverse software' is used in difficult chains of a redundant computer system, there might be some commonality in the software approaches which could give rise to CCF. Errors in the common specification, for example.

When CCF's do not occur exactly at the same tine, precautions can be taken by means of comparison methods between the redundant chains which should lead to detection of a failure before this failure is common to all chains. CCF analysis should take this technique into account.

Conditions

A thorough understanding of the system to be analysed.

Major Advantages

Covers areas not addressed by other techniques.

Problems or Disadvantages

Problems arise from the fact that CCF analysis cannot be formalised, but always need human creativity to identify CCF-situations. A resulting disadvantage is the man-hours effort involved in striving for completeness.

Result.

Identification of potential CCF's together with quantification where possible (e.g. by using beta factors) and qualification of their effect. From the quantification, the calculated frequencies of common mode failures would be assessable against criteria, if existing, and cost of CCF-effects and counter-measure costs can be balanced. Assessment of defences employed against CCF.

Relationship to Other Methods

A necessary supplement to the other methods. It profits from a previous Event Tree, or Cause Consequence Analysis, as well as a Failure Modes and Effects Analysis in that a thorough understanding of the system is a prerequisite to CCF Analysis. Identified potential CCF can be further evaluated in these techniques. Probabilistic methods are applicable whenever CCF rates can be determined. In cases where CCF's do not occur simultaneously, probabilistic methods for individual failures can be a substitute to CCF Analysis.

References.

1 Bibliography by B.S. Dhilon, Microelectronic Reliability. Vol. 18, Pergamon Press (1979) 0026-2714 78 0533 502.00/0.

2. Review of Common Cause Failures. - I.A. Watson, UKAEA, Centre for Systems Reliability, Wigshaw Lane, WA3 4NE, England, NCSR R 27, July 1981.

3. Common-Mode Failures in Redundancy Systems. I.A. Watson and G.T. Edwards Nuclear Technology Vol. 46, De. 1979.

4. Programmable Electronic Systems in Safety Related Applications; Health and Safety Executive, Her Majesty's Stationary Office.

B.3.5.3 Markov Models

Aim

To evaluate the reliability, safety or availability (ie., dependability attributes) of a system.

Description

A graph of the system is constructed. The graph represents the status of the system with regard to its failure states (the failure states are represented by the nodes of the graph). The edges between nodes, which represent the failure events or repair events, are weighted with the corresponding failure rates or repair rates. Note that the failure events, states and rates can be detailed in such a way that a precise description of the system is obtained, e.g. detected or undetected failures, manifestation of a larger failure etc.

The Markov technique is suitable for modelling redundant systems in which the level of redundancy varies with time due to component failure and repair. Other classical methods, for example, FMEA and FTA, cannot readily be adapted to modelling the effects of failures throughout the life-cycle of the system since no simple combinatorial formulae exist for calculating the corresponding probabilities.

In the simplest cases, the formulae which describe the probabilities of the system are readily available in the literature or can be calculated manually. In more complex cases, some methods of simplification (absorbing states) exist. For very complex cases results can be calculated by computer simulation of the graph.

Conditions

Knowledge of the rates (e.g. failure rate, repair rate, etc), or at least conservative estimates. For more complex cases, an appropriate simulation program which would have to be obtained.

Major Advantages

Markov Modelling provides an easy way to estimate the dependability of systems in which the level of redundancy varies during the life-cycle of the system.

Parametric studies can be performed to assess the impact of:

- different system configurations
- different rates
- different repair strategies.

Problems or Disadvantages

Hard to solve analytically, a Markov simulation is necessary for most practical applications

Depending on the simulation method of the program, some results may not be truly representative when the rates differ by orders of magnitude

The number of states may become very large.

Relationships to Other Methods

The method is probabilistic. It can be used as a complement to other analysis methods: Failure Modes and Effects Analysis, Fault Tree Analysis, Cause-Consequence Diagrams.

References

1. The Theory of Stochastic Processes. R.E. Cox and H.D. Miller.
Methuen and Co. Ltd London, Uk, 1968

2 Finite MARKOV Chains. - J.G. Kemeny and J.L. Snell.
D. Van Nostrand Company Inc. Princeton. 1959

3. Reliability Handbook B.A. Koslov and L.A. Ushakov.
Holt Rinehart and Winston Inc. New York 1970

4. The Theory and Practice of Reliable System Design.
D.P. Siewiorek and R.S. Swarz.
Digital Press 1982.

B.3.5.4 Reliability Block Diagram

See Clause B.3.4.8 of this Annex.

B.3.6 Static Analysis

B.3.6.1 Boundary Value Analysis

See Clause B.3.1.1 of this Annex.

B.3.6.2 Checklists

See Clause B.3.3.2 of this Annex.

B.3.6.3 Error Guessing

See Clause B.3.1.2 of this Annex.

D.3.6.4 Fagan Inspections

Aim

Revealing errors in all phases of the program development.

Description

A 'formal' audit on quality assurance documents aimed at finding errors and omissions. The inspection process consists of five phases; Planning, Preparation, Inspection, Rework and Follow up. Each of these phases has its own separate objective. The complete system development (specification, design, coding and testing) must be inspected.

Conditions

An independent inspection organisation must be available.

Major Advantages

- Very effective error discovering method (up to 80 % of all errors).
- Checklists on what types of errors to look for are developed.
- Comprehensive statistics on classes of errors.
- Finds all classes of systems defects concerning planned quality (not just logic errors).
- Enhances 'Ego-less' programming.
- All documentation is more readable.

Problems or Disadvantages

- The inspectors must be independent of the programmers.
- The programmers may feel the inspection is an attack on their character or work.
- 3 to 8 people, including a moderator, are needed for a proper inspection.

Results

- Inspection check lists.
- Error statistics.

Relationship to Other Methods - Complementary to testing.

Reference

1. Design and Code Inspections to Reduce Errors in Program Development. - M.E. Fagan.
IBM Systems Journal. No. 3. 1976.

B.3.6.5 Formal Design Reviews

A formal design review is an advisory activity whose primary purpose is to provide ver Maintainability, TC56, have published a Guide on Formal Design Reviews, which includes a general description of formal design reviews, their objectives, details of the various design review types, the composition of a design review team and their associated duties and responsibilities. The IEC document also provides general guidelines for planning and conducting formal design reviews, as well as specific details concerning the role of independent specialists within a design review team. Examples of specialist functions include, amongst others, Reliability, Maintenance Support and Availability.

The IEC recommend that a "formal design review shall be conducted for all new products/processes, new applications, and revisions to existing products and manufacturing processes which affect the function, performance, safety, reliability, ability to inspect maintainability, availability, ability to cost, and other characteristics affecting the end product/process, users or bystanders".

References

1. Reliability and Maintainability Guide on Formal Design Review (Draft).
International Electrotechnical Commission,
Technical Committee No 56, January 1988.

B.3.6.6 Formal Mathematical Modelling

See Clause B.3.4.3 of this Annex.

D.3.6.7 Formal Proof of Program

Aim

Using theoretical and mathematical models and rules it is possible to prove the correctness of a program without executing it.

Description

A number of assertions are stated at various locations in the program, and they are used as pre- and post conditions to various paths in the program. The proof consists of showing that the program transfers the preconditions into the postconditions according to a set of logical rules, and that the program terminates.

Conditions

Assertions must be given.

Major Advantages

- Rigorous statement on correctness achievable.
- Possibly the only way of showing the correctness of general WHILE loops.

Problems and Disadvantages

- Probably only practical in the verification of simple program modules with limited number of paths.
- No formalism available for finding complete or correct loop assertions or loop invariants.
- Difficult to provide assertions and verify their correctness.

Results

Correctness of the program with respect to the assertions.

Relationship to Other Methods

The method can be used together with program analysis.

References

1. See Clause D.3.1 for a description of Formal Methods.
2. Can Program Proving be made Practical. - J. Dahl. Research Report, ISBN 82-90230-26-5-No 33, Oslo, May

B.3.6.8 Sneak Circuit Analysis

Aim

To detect an unexpected path or logic flow within a system which, under certain conditions initiates an undesired function or inhibits a desired function

Description

A sneak circuit path may consist of hardware, software, operator actions, or combinations of these elements. Sneak circuits are not the result of hardware failure but are latent conditions inadvertently designed into the system or coded into the software programs, which can cause it to malfunction under certain conditions.

Categories of sneak circuits are:

- Sneak Paths which cause current, energy, or logical sequence to flow along an unexpected path or in an unintended direction.
- Sneak Timing in which events occur in an unexpected or conflicting sequence.
- Sneak Indications which cause an ambiguous or false display of system operating conditions, and thus may result in an undesired action by the operator.
- Sneak Labels which incorrectly or imprecisely label system functions, e.g. system inputs, controls, displays, buses, etc., and thus may mislead an operator into applying an incorrect stimulus to the system.

Sneak circuit analysis, relies on the recognition of basic topological patterns with the hardware of software structure (e.g. six basic patterns are identified for software). Analysis takes place with the aid of a checklist of questions about the use and relationships between the basic topological components.

Conditions

The basic design must be available, and broken down into is topological components, an adequate set of questions should be available.

Major Advantages

Applicable to hardware, software and the combined system. Can be applied to programs written in any language. Directed towards system design flaws rather than component failures.

Problems or Disadvantages

Labour intensive. Likely to be performed late in the development cycle, therefore changes will be expensive.

Relationship to other methods

Complementary to FMECA and FTA.

References:

1. Sneak Analysis and Software Sneak Analysis.
   S.G. Godoy and G.J. Engels. J. Aircraft Vol. 15. No. 8. 1978.
2. Sneak Circuit Analysis. - J.P. Rankin.
   Nuclear Safety. Vol. 14. No. 5. 1973.

B.3.6.9 Walkthroughs

Aim

The revealing of errors in all phases of the program development.

Description

A walkthrough team selects a small set of paper test cases, representative sets of inputs and corresponding expected outputs for the program. The test data is then manually traced through the logic of the program.

Conditions

An independent walkthrough team must be available.

Major Advantages

- Identifies the most error-prone sections of the program thus allowing one to focus more attention on these sections during the testing process.
- Very effective in finding logic design and coding errors.
- Enhances 'Ego-less' programming

Problems or Disadvantages

The programmer may feel the walk through is an attack on his character or work.

Results

A precise description of the nature of programming errors.

Relationship to Other Methods

Similar to program (Fagan) inspections.

References

1. The Art of Software Testing. - G. Myers.
Wiley & Sons, New York, 1979.

B.3.7 Quality Assurance

B.3.7.1 Checklists

See Clause B.3.3.2 of this Annex.

B.3.7.2 Fagan Inspections

See Clause B.3.6.4 of this Annex.

B.3.7.3 Formal Design Reviews

See Clause B.3.6.5 of this Annex.

B.3.8 Reliability

B.3.8.1 Failure Modes, Effects and Criticality Analysis

See Clause B.3.3.4 of this Annex.

B.3.8.2 GO Charts

See Clause B.3.4.4 of this Annex.

B.3.8.3 Reliability Block Diagrams

See Clause B.3.4.8 of this Annex.

D.3.8.4 Reliability Growth Models

Aim

To predict the current software failure rate and hence the operational reliability.

Description

Two kinds of models are to be distinguished here:

- models which estimate the number of remaining errors in a software product, and hence minimum time to correct these bugs. These models are known as Musa models, Jelinsky- Moranda models or Bug-counting models;
- models which provide the current failure rate of a program, and hence minimum time required to reach a certain reliability. The most famous examples are the Littlewood and the Littlewood-Verrall models.

In the first case, the failure rate of the software is assessed to be a function of the number of remaining errors (e.g. a sum of gamma functions for Littlewood model): the aim is to estimate the corresponding parameters.

Conditions

Development history of the particular software is required.

Major Advantages

Means of obtaining a quantitative opinion about the results of the software development phase.

Problems or Disadvantages

The diversity of reliability models prevents users from having contractual confidence in the results. Moreover, the usual assumptions made in order to run these models (independence of faults, randomness of test samples, no addition of bugs when correcting, etc) are too far removed from the real software development conditions.

References

1. Evaluation of Competing Software Reliability Predications.
   A.G. Abdalla, P.Y. Chan, and B. Littlewood.
   IEEE Trans Software Engineering, Vol SE-12, No.9 Sept 1986.

2. Software Reliability Status and Perspectives.
   C.V. Ramamoorthy and F. Bastini.
   IEEE Trans Software Engineering, Vol SE-8, 1982.

3. Software Reliability Models : A Review, - J.G. Shantikumar. Microelectr.
   Reliability Vol 23 No. 5 1983.

4. Dependability Evaluation of Software Systems in Operation.
   J.C. Laprie, IEEE Trans Software Engineering Vol. SE-10 No.6, Nov. 1984.

[INFORMATIVE]

ANNEX C

Documentation

C.1 Introduction

The documents necessary for the production of safety related software are listed in Clause 12.2 of this International Standard and detailed in figure C1 - Document Overview. This Annex gives details of the minimum points which each of the documents should address.

C.2 Software Quality Plan

The Software Quality Plan addresses all phases of the life cycle of the software. It interprets the quality requirements of this International Standard and of ISO 9001 for application to the software under development.

If a software project is part of a project covered by a System Quality Plan then a separate quality plan for software may be considered unnecessary. In this case, with the approval of the Authority Manager for the project, the provisions relevant to software shall be covered in the System Quality Plan and the Software Quality Plan omitted.

C.2.1 Life Cycle

The life cycle applied to the software shall not conflict with the life cycle described in Annex E. Any relaxations of the controls on the life cycle shall be detailed including the nature, the reasons and the justification for the relaxation.

C.2.2 Documentation

The documentation to be produced for the project shall be specified. This list of documents shall be complete, enabling a check to be made at any stage of the project that no documents have been omitted.

All of the documentation listed in Clause 12.2 of this International Standard shall be listed, unless certain documents are not required due to the relaxation of the controls or the life cycle (Clause C.2.1).

There shall be no project documents which cannot be traced to this document list.

C.2.3 Authorisations/Responsibilities

This section should specify the individual(s) responsible for authorising:

- the start of each life cycle phase,
- modifications to the specifications,
- modifications to tested code,
- test completion,
- retesting,
- acceptance of non conforming software,
- verification and validation acceptance,
- unscheduled reviews,
- Quality Plan relaxations.

C.2.4 Errors/Corrections

This section shall describe the procedures used for problem and deficiency reporting and correction. The procedures shall detail the means for:

- reporting and recording problems,
- analysis of problems and trends,
- review and testing of corrections,
- reviewing and tracking reported problems,
- controlling and accepting non conforming software.

C.2.5 Verification, Validation and Reviews

This section shall describe the degree of independence required for each scheduled verification, validation or review and for any unscheduled reviews. The degree of independence shall be compatible with those described in Clause 13 and Annex A, Clause A.5.

C.2.6 Configuration Management

This section specifies the procedures to be used for identification, documentation and control of all software with which the project is concerned.

C.2.7 Software Tools/Proprietary Software

This section specifies any pre-existing software to be used. The means used to validate each such piece of software shall be described, referencing any supporting documentation and taking account of the requirements of Annex D

Software tools constructed specifically for the project shall be subject to the same controls as deliverable software. This section shall specify any relaxation of the controls authorised on non-deliverable tools.

C.2.8 Dispatch/Delivery

This section describes the procedures for marking, handling, storing, preserving, packing and shipping the deliverable software.

C.2.9 Installation

This section describes the procedures for installing and putting into operation the delivered software. If necessary first installation and reinstallation after any service actions are to be distinguished.

C.2.10 Software Safety Plan

This section describes the procedures undertaken to ensure that the software is safe. This is detailed in Clause 4.1.

C.3 Software Requirements Specification

The Software Requirements Specification describes the function of the software and its structure as far as this is dictated by its requirements.

Over specifying should be avoided.

C.3.1 General

C.3.1.1 Purpose of the Software

The overall purpose of the software should be stated in terms of its effects on the equipment it aims to control. In as far as it is practicable, this shall also include a description of what the software should not do. The relationship between these objectives and existing policies and traditional practices should also be indicated.

The performance goals of the system – both intrinsic (such as throughput and accuracy) and extrinsic (such as economic) – are detailed. These should be ranked by priority where appropriate. For each one there is an explicit statement of whether it is absolutely necessary for the software to be acceptable for its purpose or whether it is a goal to be aimed for.

C.3.1.2 Function of the Software

The function of the software is stated in terms of the transformations between its inputs and outputs. The correspondence between these and the stated objectives and constraints should be indicated. The inputs and outputs should be identified explicitly and a preliminary description of them given. The functions should be described in behavioural terms rather than in procedural ones

These functional requirements are further refined. Input and output handling functions should be derived together with any necessary validation functions (such as format, field or logic checks). Any necessary data base maintenance, updating and information retrieval functions are specified.

Related data elements are grouped and the group activity relationship (relationship between each data group and the inputs and outputs of the software) determined. Any groups with special requirements are identified.

Finally, all the application-specific calculations or application-dependent data manipulations that must be carried out are fully specified, preferably with the help of diagrams depicting functional and I/O relationships.

C.3.1.3 Compatibility with other Software Systems

The information exchanges between the software and any other software systems are identified and described:

- input and output communication ports
- communications protocols and procedures
- exception handling and error recovery
- message formats
- message throughput

C.3.1.4 Imposed Constraints

Constraints or requirements on the use or non-use of specific technical solutions are listed.

Such constraints can include:

- existing or imposed hardware or software
- supply availability (both present and future)
- facilities available

Some software approaches that will satisfy the software system capability requirements may also be listed. These software approaches may be presented in the form of actual physical configurations, but preferably without reference to any specific equipment manufacturer. Any solution offered should be presented in sufficient detail to be understood by the supplier without further supplementary information. In any case, the identification of these approaches should not preclude the suggestion of others by the supplier.

C.3.1.4.1 *Performance Constraints*

This section should specify both the static and the dynamic numerical requirements placed on the software or on human interaction with the software, as a whole.

Static numerical requirements may include:

- The number of terminals to be supported.
- The number of simultaneous users to be supported.
- Number of files and records to be handled.
- Sizes of tables and files.

Dynamic numerical requirements may include, for example the number of transactions and tasks and the amount of data to be processed within certain time periods for both normal and peak workload conditions.

All of these requirements should be stated in measureable terms.

C.3.1.4.2 Design Constraints

Design constraints can be imposed by other standards, hardware limitations etc.

There shall be a subsection specifying requirements derived from existing standards or regulations. These might include:

- Report format.
- Data naming.
- Accounting Procedures.
- Audit Tracing.

There shall also be a subsection specifying the hardware limitations. This subsection could include requirements for the software to operate within various hardware constraints. For example these could include:

- Hardware configuration characteristics (number of ports, instruction sets, etc.)

- Limits on primary and secondary memory.

C.3.1.4.3 Attributes

There are a number of attributes that can place specific requirements on the software. Some of these are indicated below. These should not be interpreted to be a complete list.

Availability

This paragraph could specify the factors required to guarantee a defined availability level for the entire system such as checkpoint, recovery and restart.

Security

This could specify the factors that would protect the software from accidental or malicious access, modification, destruction or disclosure. Specific requirements in this area could include the need to:

- Utilize certain cryptographical techniques.
- Keep specific log or history data sets.
- Assign certain functions to different modules.
- Complete checksums for critical quantities
- Restrict communications between some areas of a program.

Maintainability

This paragraph could specify the requirements to ensure that the software could be maintained. For example, as part of this,

- Specific coupling metrics for the software modules could be required

- Specific data/program partitioning requirements could be specified for micro-devices.

Transferability/Conversion

This paragraph could specify the user procedures, user interface compatibility constraints (if any) etc. required to transport the software from one environment to another.

Caution It is important that required attributes be specified so that their achievement can be objectively verified by a prescribed method.

[illegible] External Interface Requirements

User Interfaces

This paragraph should specify the characteristics that the software must support for each human interface to the software product. For example, if the user of the system operates through a display terminal, the following should be specified:

- Required screen formats
- Page layout and content of any reports or menus
- Relative timing of inputs and outputs
- Need for some form of programmable function keys

This paragraph should also specify all the aspects of optimizing the interface with the person who must use the system. This may simply comprise a list of do's and don'ts on how the system will appear to the user.

Hardware Interfaces

This paragraph should specify the logical characteristics of each interface between the software product and the hardware components of the system. It also covers such matters as what devices are to be supported, and protocols.

Software Interfaces

This paragraph should specify the use of other required software products and interfaces with other application systems.

For each required software product, the following should be provided:

- Name
- Mnemonic
- Specification number
- Version number
- Source

For each interface, this part should:

- Discuss the purpose of the interfacing as related to this software product
- Define the interface in terms of message content and format.

It is not necessary to detail any well-documented interface, but a reference to the document defining the interface is required.

Communications Interfaces

This paragraph should specify the various interfaces to communications such as local network protocols etc.

C.3.1.4.1 Other Requirements

Certain requirements may, due to the nature of the software, the user organization etc. be placed in separate categories as indicated below.

Data Base

This paragraph could specify the requirements for any data base that is to be developed as part of the product. This might include:

- Frequency of use.
- Accessing capabilities.
- Data element and file descriptors.
- Relationship of data elements, records and files.
- Static and dynamic organization.
  Retention requirements for data.

Operations

This paragraph could specify the normal and special operations required by the user such as:

- Various modes of operations in the user organization; for example, user-initiated operations.
- Periods of interactive operation and periods of unattended operation.
- Data processing support functions.
  Backup and recovery operations.

Site Adaptation Requirements

This paragraph should define the requirements for any data or initialization sequences that are specific to a given site, mission, or operational mode, for example grid values, safety limits etc.

Specify the site related features that should be modified to adapt the software to a particular installation.

C.3.1.5 Software Structure

The Software Requirements Specification should distinguish each functional part of the software system and define each function clearly. Furthermore, each software subsystem so distinguished should have its inputs identified, its communication requirements with other software subsystems detailed (procedures and protocols, messages, formats, etc.) and any other specific technological requirement specified.

C.3.1.6 Managerial Details

This section identifies:

man-machine interfaces and any related potential problems

operational procedures together with any potential unsafe actions and the possible causes of them

human procedures within the software system which are likely to be bottlenecks, or are likely to cause special problems to the software system, such as operations which could give rise to many errors

For each of these areas all operational, ergonomic and other engineering factors having a significant effect on human performance should be determined (for example alternative hardware designs of video display units which would influence the manual operation of the display). The human procedures that will require significant training or elaborate aids should be identified. More detailed standards should be followed where these are available and appropriate

C.3.1.7 Documentary Details

This section states all the relevant documentation that is to be produced for the software system components. Notice that this documentation should cover the whole of the software life cycle and include the operational and maintenance requirements.

C.3.1.8 Reliability and Safety

This covers all areas of failure and error handling. Requirements are typically to be specified for:

Software failure:
- failure detection,
- failure identification,
- failure isolation,
- remedial procedures,
- reliability requirements,
- redundancy,
- failsafe operation.

Recovery Procedures:
- fallback procedures,
- reconstruction of obliterated or incorrectly altered data.

Information Collection:
- different sources,
- different channels.

Information Processing:
- weak signals,
- data storage,
- analysis techniques,
- noise reduction,
- error handling.

Human Error or Violations:
  - security standards.
  - control aspects.
  - procedural aspects.
  - management aspects.
  - research aspects.

C.3.2 External Requirements

This section contains those requirements on the external equipment and procedures which are necessitated by the introduction of the software. The requirements for these changes should be kept well separated from the 'real' requirements.

C.3.2.1 Functional Requirements

This section should identify those adjacent and interacting software systems whose interface parameters are to be specified. These interfaces will typically involve input data sources and output data links.

C.3.2.2 Physical Environment

This section lists those physical factors which may have no apparent relevance to the function of the software system but which potentially influence the function nevertheless. These typically include the environmental factors which the software system must be designed to tolerate.

C.3.3 Project Requirements

The project is introduced by describing briefly the philosophy underlying it, identifying the relevant departments/bodies involved in it and detailing their respective role, policies and commitment towards it.

C.3.3.1 Purpose and Function of the Project

The many interrelated objectives of the Project are stated and correlated. Some of these fall into classes including:

System Related:
  - identification.
  - operation.
  - safety and reliability.

Organisational:
  - planning and scheduling.
  - production.

Managerial:
  - responsibility and competence.
  - control and interaction.
  - productivity.

Quality Assurance:
  - testing.
  - acceptance criteria.
  - certifications needed.
  - change procedures.

C.3.3.2 Compatibility with other Software/Systems

The position of the Project with respect to other relevant software or systems in development is indicated.

Any other developments related to the life cycle for the software in question are also referenced

C.3.3.3 Selection of Technology

The main aspects of the technology selected for the project are stated together with their respective requirements and quality attributes. These may be related to:

- Persons involved.
- Information system.
  Data collection, storage and distribution.
- Production tools:
  - hardware components,
  - standards,
  - development aids,
  - compilers,
  - debuggers.
- Test and verification tools:
  - simulators,
  - signal generators,
  - in-circuit emulators,
  - performance measurement techniques,
  - software verification tools.

C.3.3.4 Main Components

The main procedural units of the project are defined according to organisational, functional, geographic, or time dependent aspects. These may be correlated with material availability, sources, production responsibility etc. A plan is drawn up showing the flow of information, material and control between these units.

C.3.3.5 Documentation Details

This section details design and engineering standards, documents relevant to the documentation plan and quality assurance plans. These should be cross referenced.

C.3.3.6 Safety and Reliability

In as much as the development process may involve aspects of safety and reliability, these are mentioned here. Such hazards may include dangers to the developers in operating equipment during the development or maintenance phases (eg. eye strain or excessive X-ray dosage due to prolonged usage of visual display units).

C.3.3.7 Maintenance

Consideration should be given to the organisation, persons, training and equipment needed to keep the development process viable during the time range of the complete software life cycle.

C.3.4 Software Acceptance Test Specification

The Software Acceptance Test Specification does not form part of the Software Requirements Specification, but is developed in parallel with it. The format of the Software Acceptance Test Specification is described in Clause C.6

This section of the Software Requirements Specification document describes any aspects of the Software Requirements Specification which require special consideration in the Software Acceptance Test Specification.

C.3.5. Definitions Used

All technical terms and any common terms used with a 'special' meaning in the Software Requirements Specification are defined.

C.3.6 References

This section contains a resume of all references to sources outside the Software Requirements Specification. It should also contain cross-reference lists showing where defined words are used, synonyms, etc.

C.4 Software Design Specification

C.4.1 General

The Software Design Specification describes the function and structural breakdown of software being used to meet the Software Requirements Specification. This is the first level at which other than functional description is detailed.

Technical software terminology may be used in the descriptions.

C.4.1.1 Purpose

The purpose of the software should be briefly described, together with the intended readership for the Software Specification.

C.4.1.2 Scope

This section should:

- Identify the software product(s) to be produced by name.
- Explain what the software product will, and if necessary will not, do.
- Describe the application of the software being specified. As a portion of this it should:
  - Describe all relevant benefits, objectives and goals as precisely as possible.
  - Be consistent with similar statements in the Software Requirements Specification.

This information may be provided by reference to the relevant part of the Software Requirements Specification.

C.4.1.3 Definitions

This subsection should provide definitions of all terms, acronyms and abbreviations required to properly interpret the Software Design Specification. This information may be provided by reference to an appendix, or appendices.

C.4.1.4 References

This subsection should:

- Provide a complete list of all documents referenced.
- Identify each document by title, report number (if applicable), date and publisher.
- Specify the sources from which the references can be obtained.

This information may be provided by reference to an appendix or other document (eg System Specification).

C.4.1.5 Overview

This subsection should:

- Describe what the Software Design Specification contains.
- Explain how the Software Design Specification is organised.

C.4.2 General Description

This section should describe the general factors which affect the software and its requirements. The intention should be simply to clarify the requirements, not to state specific requirements.

C.4.2.1 Relationships with other Software and Systems

This subsection puts the software into perspective with other related software and with the system of which the software forms a part.

Software cannot be independent and totally self contained. As a minimum the hardware on which the software is required to run, and the document(s) which specify it, must be stated here.

If the Software Design Specification defines a product which is a component of a larger system or project, then this subsection should:

- Describe the functions of each component of the larger system, and identify any interfaces with this software.
- Identify the principal external interfaces of this software product.

C.4.2.2 Functions

This subsection should provide a summary of the functions that the software will perform. It may be possible to take this directly from the Software Requirements Specification.

C.4.2.3 User Characteristics

This subsection should describe those general characteristics of the eventual users of the product that will affect the specific software design.

C.4.2.4 General Constraints

This subsection should provide a general description of any other items that will limit the software engineer's options for designing the system. These can include:

- Regulatory policies,
- Interfaces to hardware and other software,
- Parallel operation,
- Audit functions,
- Control functions,
- Higher-order language requirements,
- Software handshake requirements,
- Criticality of the application,
- Safety and reliability considerations.

C.4.2.5 Assumptions and Dependencies

This subsection should list each of the factors that affect the requirements. These factors are NOT design constraints on the software but are, rather any changes to them that can affect the requirements.

C.4.3 General Design

This section should:

- Describe a general design structure which will be used to fulfil the specific requirements laid out in the Software Requirement Specification.
- Demonstrate that the general structure described provides a complete and consistent realisation of the requirements and constraints laid out in the Software Requirements Specification.

C.4.3.1 Module Structure

This specifies the breakdown of the software system into modules.

A module, which is a single functional unit of code, should be defined more specifically for a particular software system. A size should be specified, which may be lines of source code, bytes of source or executable/object code, etc. The module definition must be appropriate to the overall function and expected size of the completed software system.

C.4.3.2 Structure Overview

This subsection should describe the global structures to be used in the software implementation.

a) Types - That is the data types which must be recognised throughout the software.

b) File Structures - That is the structure of individual files and the distribution of data between files.

c) Data Structures - The structure of data in computer memory, which is used by all parts of the system, and its relationship to the file structures.

d) Available Libraries - The code libraries which may called on implicitly or explicitly by the software.

e) Interfaces - The interfaces with hardware and other software and the interfaces between the constituent modules of this software. Use of global variables, directly accessed memory areas, etc. should be specified here.

C.4.3.3 Flowchart/Formal Description

This subsection should formally describe the overall structure of the software. The formal description should use a method appropriate to the application. If a method is not specified in the Software Requirements Specification then a suitable one should be chosen from the methods detailed Annex A.

The constituent modules which make up the software and their interrelationship should be formally described.

C.4.3.4 Structure/Function Relationship

This subsection should describe the means by which the structures described in section C.4.3.2 are to be used to fulfil the functional requirements identified in the Software Requirements Specification

An overview of the software structure and its relationship to the requirements shall be described.

For each module, the function of that module should be specified, together with those items of global data on which it operates, or to which it has access. Any global routines or libraries which the routine may NOT access must be specified.

C.4.3.5 Structure/Performance Relationship

This subsection should describe how the structures specified here meet the static and dynamic numerical requirements laid out in the Software Requirements Specification (Clause C.3.1.4.1).

C.4.3.6 Structure/Constraints Relationship

This subsection should describe how the structures specified here meet the standard compliance and hardware limitation requirements laid out in the Software Requirements Specification (Clause C.3.1.4.2).

C.4.3.7 Structure/Attributes Relationship

This subsection should describe how the structures specified here meet the requirements regarding the attributes of availability, security, maintainability, transferability etc laid out in the Software Requirements Specification (Clause C.3.1.4.3).

C.4.3.8 Structure/External Interface Relationship

This subsection should describe how the structures specified here meet the requirements regarding user, hardware, software and communications interfaces laid out in the Software Requirements Specification (Clause C.3.1.4.4).

C.4.3.9 Structure/Other Requirements Relationship

This subsection should describe how the structures specified here meet the requirements regarding data base, operations and site adaption laid out in the Software Requirements Specification (Clause C.3.1.4.5).

C.4.4 Module Requirements

This section should provide an overview of the requirements placed on each module by the interaction of the structures chosen and the overall requirements.

C.4.4.1 Common Variables/Routines

This subsection should identify the global variables and routines which are accessible by all modules which make up the software.

C.4.4.2 Libraries Available

This subsection should identify those software libraries which are available to the software.

C.4.4.3 For each Module

This section provides the basis for the individual Software Module Specifications (Clause C.5), for each software module it should briefly:

- Detail the requirements which are placed on the module by the relationships in Clause C.3.3.
- Describe the function of the module.
- Describe the Entry and Exit conditions which the module is required to satisfy.
- Describe the failure conditions which the module is required to deal with.
- List any requirements regarding register preservation.
- Reference any mathematical formulae or principle which the module is required to use.
- List any special requirements regarding module size.

C.4.5 Software Integration Test Specification

The Software Integration Test Specification does not form part of the Software Design Specification, but is developed in parallel with it. The format of the Software Integration Test Specification is described in Clause C.6.

This section of the Software Design Specification document shall describe any aspects of the Software Design Specification which require special consideration in the Software Integration Test Specification.

C.5 Software Module Specification

The Software Module Specification shall contain a brief summary of the module and its functions.

This shall be followed by the following sections:

C.5.1 User Information

The user information shall contain information pertaining to;

- entry and exit conditions,
- code insertions,
- failures and failure conditions which are detected and reported.
- registers used.

Generation details of the module shall be included, that is a specification of how the module is generated within the subsystem with any options used.

C.5.2 Data Interfaces

The data interfaces shall specify any absolutes, local variables and global variables used.

The names of any libraries, whether explicitly named or implicitly used, shall be identified as well as any common re-entrant procedures used by the module.

C.5.3 Program Description

The program description shall;

- state any mathematical formulae or principles used.
- contain a written and unambiguous description of the module.
- specify all resources required by the module at run- time minimum and maximum number of disc accesses.

C.5.4 Flowchart, Structogram or Pseudo Code

Flowchart at module level, to be supported by listings with extensive comments.

C.5.5 References

Reference to all relevant documentation shall be made.

C.5.6 Software Module Test Specification

The Software Module Test Specification does not form part of the Software Module Specification, but is developed in parallel with it. The format of the Software Module Test Specification is described in Clause C.6.

This section of the Software Module Specification document describes any aspects of the Software Module Specification which require special consideration in the Software Module Test Specification.

C.6 Test Specifications

This clause applies to the Software Acceptance Test Specification, the Software Integration Test Specification and the Software Module Test Specification.

This document shall be developed in parallel with the appropriate specification for the life cycle phase and shall give detailed information on the tests to be performed in order to systematically test all sections of that specification (see Annex E).

The test specifications shall include the environment in which the tests are to be run. Error detection and corrective action procedures shall be specified if they differ from those in the Quality Plan (see Clause C.2.4).

C.7 Test Reports

Software Test Reports are produced for the Software Acceptance Test, the Software Integration Test and the Software Module Tests. The Software Test Reports shall present the results of the test performed as detailed in the relevaent test specification. The Software Test Reports shall be issued whether or not the software has achieved the performance requirements given in the Software Test Specifications. This International Standard shall allow the complete diagnosis of design and performance deficiencies. It shall also include a description of the process of resolution of all software deficiencies and test discrepancies discovered during the test (see Clause C.2.4).

The Software Test Reports shall include the following items for all tests carried out:

a) hardware configuration and version used for the test;
b) software configuration and version used for the test;
c) storage medium used by and access requirements of the final code tested;
d) input test data;
e) output test data;
f) additional data regarding timing, sequence of events, etc;
g) degree of conformance with acceptance criteria given in the Software Test Specification;
h) error incident log which describes the character of each error and the action taken by the design team.

C.8 Verification, Validation and Review Plans

The purpose of verification plans is to ensure that the documents produced during the phases of the life cycle adequately reflect the previous phase's specification and that the test specification for the current phase matches the specification produced for that phase (see Clause 9).

The purpose of the Software Design Review Plan is to ensure that the documents produced during the design phases of the software life cycle adequately specify both the software to be produced and the criteria on which the completed software will be judged (see Clause 9.3).

The purpose of the Software Validation Plan is to ensure that the software produced performs the requirements detailed in the Software Requirements Specification and will pass the Software Acceptance Test (see Clause 11).

The verification, validation and review plans shall detail;

- the procedures to be used,
- the personnel involved (with respect to independence),
- the documents to be used,
- the documents to be produced,
- the methods which will be used to monitor the effectiveness of the exercise

C.9 Verification, Validation and Review Reports

The reports shall present the results of the verification, validation or review performed as detailed in the appropriate verification, validation or review plan for the life cycle phase. The report shall be issued whether or not the specified requirements given in the verification, validation or review plan have been met.

The report shall include the following;

- the procedures used,
- the personnel involved,
- the date performed,
- the documents used,
- the documents produced,
- the methods used to monitor the effectiveness of the exercise,
- the effectiveness of the exercise.

C.10 Source Code

The program source code listing will contain the following minimum information.

Header

a) Author - the original author of the program
b) Identity - the programming and associated specification
c) Contract - the contract/project code
d) Origination - the date of origination of the program
e) Approval State - the state of approval that the program has achieved: Development /Test /Incorporated /Released with appropriate date.
f) Version Control - a record of changes made (in brief) with date, person effecting any change and authority.

Annotation

The source shall contain such annotation as to make maintenance/modification of the code possible by persons other than the original author. Annotations are to be strategically placed throughout the code.

Variable Names

Variable names are to be meaningful and concise. Numeric variable names are to be avoided.

Comments

Comments included within the code should specify the function of any variables, describing code statements in terms of their functions rather than their actions. For example:

```
FOR Count: = 1 TO Last DO (For each record on file)
```

Version Number

The source code (Module) version number should be included as a constant in the code, so that it may be determined in the object code.

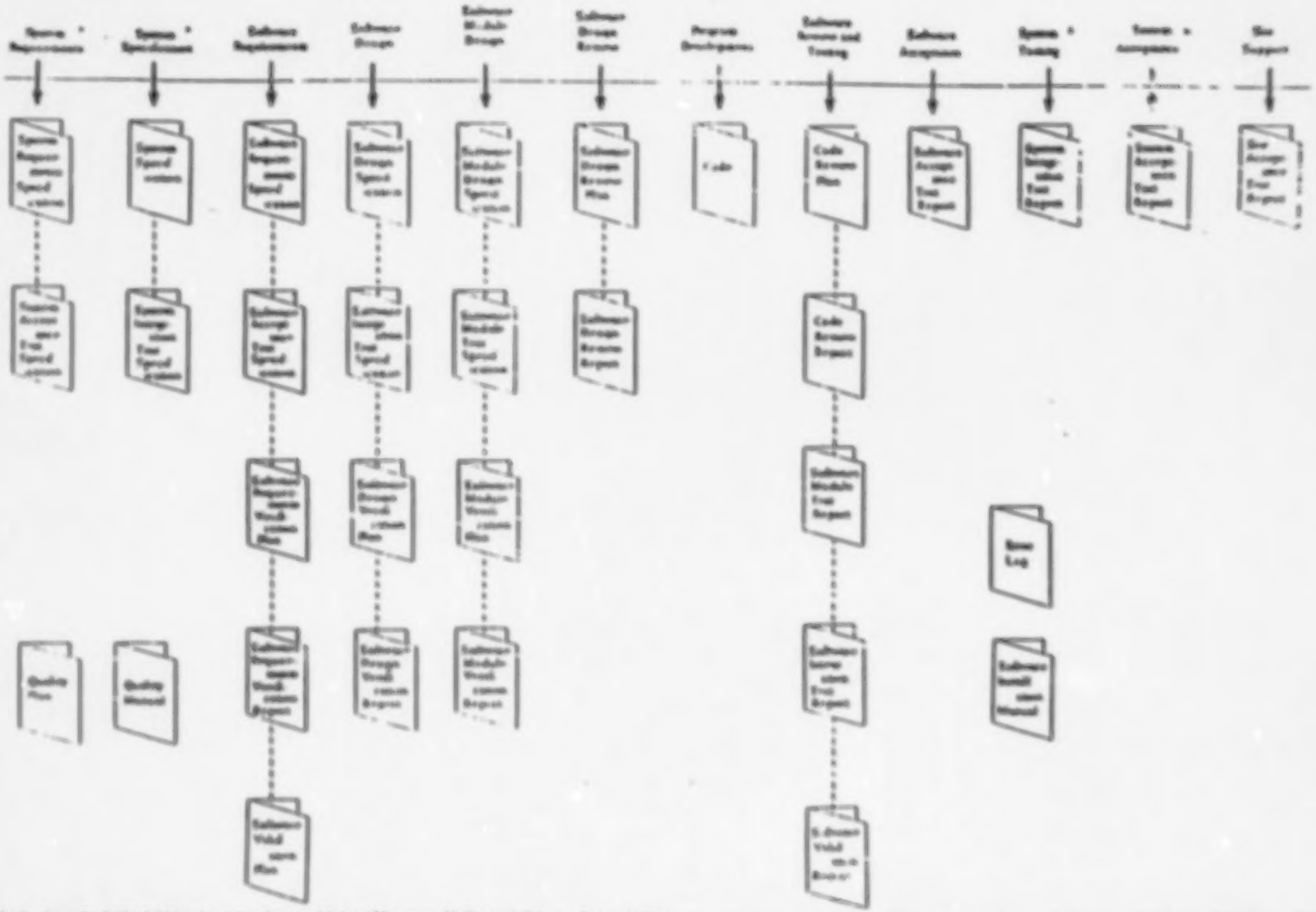
Indentation

The source code shall be indented in such a way as to emphasise the structure of the code.

Structure

Modular design using subroutines, and avoiding Jumps, shall be used.

Figure C1 [illegible]


System Requirements *
System Specification *
Software Requirements
Software Design
Software Module Design
Software Design Review
Program Development
Software Review and Testing
Software Acceptance
System Testing *
System Acceptance *
Site Support
Code

* [illegible]

65A(Secretariat)54

[INFORMATIVE]

ANNEX D

Rules for Development of Safety Related Software

D.1 Implementation of Fault Tolerant Systems

D.1.1 Fail Safe Systems

Fail-safe systems should include mechanisms to

- detect errors which may endanger safety.
- bring the system into a safe fall-back state in case of error.

For the detection of single errors some form of multiple redundancy shall be applied (internal by self checking or external by means of duplication).

Errors resulting from software design faults shall be detected by applying software design diversity. This can be realised in several ways (serial execution of two programs on a single computer, parallel execution of two programs on two computers, etc.).

In each case a two out of two voting function shall detect discrepancies between the two versions which indicate a failure in the value and/or time domain.

If such a discrepancy is detected, the system shall be brought into a safe fall-back state by an emergency switch, which can be implemented in different ways (hardware, software).

There are two requirements for fail-safe systems:

Passive : The system shall not perform actions which cause the system to enter an unsafe state.

Active : The system shall react to changes of the process in a safe manner within a required time interval.

Guidelines for Fail-Safe Systems :

- The voter and the emergency switch shall be separated from the rest of the system and should be implemented, without the use of software, using a technology in which safety characteristics, such as reliability, can be quantified.

- Two versions of a program should run on separate computers. If (for reasons of costs) only one computer is used where two versions of a program are executed serially, the minimal requirement is to have two separate I/O-areas. These separate I/O-areas are then treated as if they came from different computers.

- A cyclic signal should be provided to detect failures in the time domain.

- A maximum of design diversity should be achieved. This can be done by using different specifications, implementation teams, programming paradigms, programming languages etc.

D.1.2 Fault Compensating Systems

Because of intrinsic features of the controlled process it is not possible to react to a possibly dangerous system state by transferring to a safe fall-back state. A safe fall-back state may not exist, or the the transfer may not be possible. The system shall remain in an active (running) state, perhaps with reduced functionality. In order to achieve this the system shall mask or compensate certain faults. This covers mainly the aspect of 'fault removal' of the dependability measures.

For each method mentioned a classification (evaluation) shall be made with respect to safety related points, time constraints, additional system states (for example alert states to have time for compensation), etc.

D.1.3 Fault Detection and Diagnosis

Fault detection and diagnosis shall be applied to fault compensating systems. For details see Annex B.

D.1.3.1 Reconfiguration

Reconfiguration is one principle used in fault compensating systems to deal with faults. Since no safe fall-back state exists, a defective subsystem shall be replaced by an operating one in the event of a (permanent) failure.

This goal shall be achieved with the help of some form of active or passive redundancy, see Annex B.

D.1.3.2 Recovery

The state of any subsystem as well as of the overall system shall be checked for consistency with the external state (environment) to get reliable information about correct operation (resynchronisation with the external environment). Detection of a fault resulting in an inconsistency between internal and external states shall lead to actions to return to a consistent state. This is called recovery and should concern data recovery as well as hardware recovery.

For details about Backward Recovery and Forward Recovery see Annex B.

D.1.4 High Level Strategies

In addition to the methods used in fail-safe or fault compensating fault tolerant systems, there are some other methods on a higher strategic level where, by means of (somewhat application dependent) software, a forecast is made to detect dangerous events and to prepare for countermeasures. The high level strategies are directly mainly at 'fault forecasting' of the dependability measures.

Such methods, concurrent simulation parallel to the controlling processes, expert system support etc., can help to cope with;

- the inadequacies of a digital model modelling an analogue world
- the inadequacies of a simple model modelling the real world
- the preparation in advance of preventive actions (forecasting), without performing undesired preventive actions.

In general, such AI-based (artificial intelligence based) methods are not safer in themselves than conventional software-based methods. However by using these as a diverse approach it may be possible to cope with specification errors as a result of the radically different approach compared to traditional methods. By making use of concurrent execution and restricting the task of the AI system to check certain safety related rules, it may be possible to cope with restrictions of the conventional control systems in the time domain. The AI part allows inclusion of experience gained in the decision process. Methods like pattern recognition are often faster and simpler to use in the control process than conventional evaluation (especially in the human interface).

The following concepts shall be addressed as additional provisions to fulfill the general aims of safety related architectures;

- considering the consequences of actions (consequences prediction),
- prevention of dangerous events,
- avoidance of hazardous system states,
- detection of dangerous states in time,
- avoidance of propagation of faults.

The intention, mainly, should be to allow the system to continue to operate. As a minimum, methods of consequences prediction shall provide "alarms of inconsistencies" to the user, and should switch to a fail-safe state and/or manual operation, if this is both possible and safer.

For details of some of the methods applicable see Annex B.

D.2 Languages and Compilers

Compilers with a current national or international certificate showing conformance with the appropriate language definition shall be used. Alternatively compilers without such a certificate may be used provided that they have been thoroughly tested. When compilers which do not have a conformance certificate are used, then the adequacy of the testing performed on the compiler shall be justified in the Quality Plan.

The language chosen for developing safety related software should have the following features:

To take into account PLC's

- Strongly typed.
- Block structured.
- Formally defined syntax.
- Well defined semantics.
- Translation-time type checking.
- Run-time type and array bound checking.
- Parameter checking.

These will ensure that the languages performance and properties are not the subject of a range of different interpretations. In addition to the features detailed above, the language chosen should:

- Encourage the use of small and manageable modules.
- Allow access to certain data to be restricted to defined modules.
- Allow the definition of variable type sub- ranges.
- Allow the use of error-limiting constructs.

Where possible the use of assembly level languages should be avoided. However, there exist circumstances where the use of assembly level languages may be necessary. Such circumstances include:

- Memory constraints
- Only assembly level language available
- Performance constraints
- Customer requirements

It is further recommended that the following techniques or features should not be used in safety related software:

- Unconditional jumps in High Level Languages
- Recursion
    - Simple, to a known depth .
    - Complex, to a depth determined at run time
- Pointers or other dynamic memory usage
- Floating point arithmetic
- Interrupts
- Multiple entry or exit of
    - loops
    - blocks
    - subroutines
- Undeclared variables
- Implicit variable initialisation, assumptions regarding variables which have not been initialised.
- Complex language constructs such as
    - Variant records
    - Case statements
    - Functional parameters
    - N & 1/2 times loops

D.3 Formal Mathematical Techniques

This Clause in the Annex is intended to be a supplement for Clauses B.3.4.3, B.3.6.6 Formal Mathematical Modelling and B.3.6.7 Formal Proof to provide an indication of the different formal methods which are acailable for use and a brief description of each.

D.3.1 VDM

VDM (Vienna Development Method) is a method which enables the specification of sequential systems in terms of pre- and post- conditions of states. System data types consist of a state specification (satisfying an explicitly given invariant) together with a collection of operations. Recursively structured data types built out of sequences, sets, maps and records are also definable, together with recursively defined functions, but shall not be used for safety related software.

The syntax used should be taken from ; C.B. Jones, "Systematic Software Development using VDM", Prentice Hall International, 1986.

D.3.2 OBJ

OBJ is a specification and design language for algebraically describing software systems.

D.3.3 "Z"

"Z is a software specification technique developed at Oxford University based on Zermelo-Fraenkel set theory and predicate calculus. It is conceptually similar to the pre- and post- condition style of VDM in that system operations are described as relations on states, by logical properties. However, "Z" does not separate pre and post conditions explicitly (ie. the pre condition is left implicit as the domain of the specified relation). As in VDM, system data types may be specified, with an invariant to be satisfied. Specifications are written in terms of schema boxes, each of which contains a signature part and a predicate part. The schema calculus may then be used for combining specifications together.

The following should be used as reference; J.M. Spivey, "The Z Notation - a reference manual", Prentice Hall International, 1988.

D.3.4 HOL

HOL (Higher Order Logic) refers to a particular logic notation and its machine support system, both of which were developed at the University of Cambridge Computer Laboratory. The logic notation is mostly taken from Church's Simple Theory of Types and the machine support system is based upon the LCF (Logic of Computable Functions) system.

The definition of the current syntax should be taken from; M. Gordon, "HOL : A machine oriented formulation of higher order logic", University of Cambridge Technical Report, Number - 68.

D.3.5 CCS

CCS, the Calculus of Communicating Systems, is an algebraic method used to describe concurrent systems. CCS describes systems as a network of independent agents that communicate via ports. The inputs and outputs of which they are capable are known as actions. Operators are provided for restriction (hiding) and re-labelling.

SCCS (Synchronous CCS) has been developed from CCS to handle synchronous behaviour. It is more general than CCS in that the earlier (and simpler calculus) can be derived as a subsystem of SCCS.

The following reference should be taken as a definition of the syntax: R. Milner, "A Calculus of Communicating Systems", ECS-LFCS-86-7, Laboratory for the Foundations of Computer Science, Edinburgh University, 1986.

D.3.6 CSP

CSP, Communicating Sequential Processes, is a calculus for the description of concurrent systems. The system is modelled as a network of independent processes that communicate via channels. It is a model 'orientated' method that embodies a notion of testing equivalence.

The syntax for the method should be obtained from: C.A.R. Hoare, "Communicating Sequential Processes", Prentice Hall International, 1986.

D.3.7 Temporal Logic

Temporal logic is concerned with the description of temporal properties by using ideas and notions from model logic.

D.3.8 LOTOS

LOTOS (Language for Temporal Ordering Specification) is based on CCS, with additional features from the related algebras CSP and CIRCAL (Circuit Calculus). It overcomes the weakness of CCS in the handling of data structures and value expressions by combining it with a second component based on the abstract data type language ACT ONE. The process definition component of LOTOS could, however, be used with other formalisms for the description of abstract data types.

LOTOS is being adopted as the following ISO Draft International Standard from which the syntax shall be obtained: ISO/TC 97/SC 21, "Information processing systems - Open systems inter-connection - LOTOS - A formal description technique based on the temporal ordering of observational behaviour", ISO DIS 8807, July 20, 1987.

D.4 Automated Tools

Automated tools, if available, should be used on all stages of the software life cycle

Such tools should either have been validated by an independent third party or have been thoroughly tested and mature. When tools which have not been validated are to be used, then the adequacy of the testing performed on the tools shall be justified in the Quality Plan.

[INFORMATIVE]

ANNEX E

Software Life Cycle

E.1 Introduction

The development of software involves a number of stages designed to ensure, as far as practicable, that the interactive nature of software design continues to reflect the original objectives, and to develop a comprehensive test strategy in line with the software. This collection of stages is termed the Software Life Cycle and is embedded within the System Life Cycle.

This International Standard uses the Software Life Cycle detailed in Figure E1.

From Figure E1 it can be seen that there are 12 distinct development phases to the development of a system. 5 phases deal with the system as a whole, the remaining 7 phases (3 to 9) are split so that they deal independently with Software.

The principle behind the model is that at each stage of the design, prior to the start of the programming, a specification further defining the design is prepared along with a revision of the testing criteria.

For some systems the controls are relaxed, and life cycle phases are combined. Typically, a contract for software would involve only phases 3 to 9, provided that the software development is done on the target hardware.

E.2 System Requirements Phase

The first specification of the design is the System Requirements Specification setting out the requirements of the system as defined by the Customer. During the preparation of the System Requirements Specification the operating limits of the system will be identified and used to form the basis of the System Acceptance Test Specification. The System Acceptance Test Specification sets out the criteria to be applied to the system by the Customer to achieve acceptance.

In certain projects this phase may be combined with the System Specification phase.

Before the System Requirements Phase is considered complete the System Requirements Verification will take place. The System Requirements Specification is considered for conformity and the System Acceptance Test Specification is considered for consistency with and adherence to the testing needs of the System Requirements Specification. If both these are found to be satisfactory they are approved and the next phase may begin. Alternatively, if they are not approved then they will be subject to further work until found to be satisfactory.

E.3 System Specification Phase

On commencement of the System Specification Phase the previously prepared System Requirements Specification and System Acceptance Test Specification shall be taken as the basis of work in this phase of development.

The System Requirements Specification is used to define in further detail the design of the system as a whole and to produce the System Specification. Similarly, the System Acceptance Test Specification is refined to produce the System Integration Specification.

At the conclusion of the System Specification Phase the System Verification shall take place. The System Verification shall review:

1. The System Specification as a detailed refinement of the System Requirements Specification.

2. The System Integration Test Specification as a revision of the System Acceptance Test Specification.

3. The System Integration Test Specification as a complete and satisfactory statement of testing according to the System Specification.

E.4 Software Requirements Phase

This phase involves setting out the requirements of the software as defined by the Customer. During the preparation of the Software Requirements Specification the operating limits of the software will be identified and used to form the basis of the Software Acceptance Test Specification. The Software Acceptance Test Specification sets out the criteria to be applied to the software by the Customer to achieve acceptance

The Software Requirements Phase may be the first phase in the Software Life Cycle, either by being combined with the System Requirements Phase, or because the System Requirements and System Specification phases have been omitted. When the Software Requirements Phase is not the first Phase in the Software Life Cycle, the previously prepared System Specification and System Integration Test Specification, together with the System Requirements Specification and System Acceptance Test Specification, shall be taken as the basis for the work of the phase.

At the conclusion of the Software Requirements Phase the Software Requirements Verification shall take place to review:

1. The Software Requirements Specification as a detailed refinement of the System Specification or as an unambiguous representation of the requirements of the software.

2. The Software Acceptance Test Specification as a complete and satisfactory statement of testing according to the System Integration Test Specification.

E.5 Software Design Phase

On commencement of the Software Design Phase the previously approved System Specification and System Integration Test Specification (or Software Requirements Specification and Software Acceptance Test Specification) will be taken as the basis of work in this phase of development.

The System Specification (or Software Requirements Specification) is used to define in further detail the design of the system in terms of the software and to produce a Software Design Specification. Similarly the System Integration Test Specification (or Software Acceptance Test Specification is used to produce the Software Integration Test Specification which describes the tests required to test the software.

At the conclusion of the Software Design Phase the Software Design Verification shall take place to review:

1. The Software Design Specification as a detailed refinement of System Specification (or Software Requirements Specification).
2. The Software Integration Test Specification as a revision of the System Integration Test Specification (or Software Acceptance Test Specification)
3. The Software Integration Test Specification as a complete and satisfactory statement of testing according to the Software Design Specification.

E.6 Software Module Design Phase

On commencement of the Software Module Design Phase the previously approved Software Design Specification and Software Integration Test Specification will be taken as the basis of work in this phase of development.

The Software Design Specification is used to define the design in terms of individual Software Module Specifications. The Software Integration Test Specification is used to define the individual Software Module Test Specifications which accord with the respective Software Module Specifications.

At the conclusion of the Software Module Design Phase the Software Module Verification shall take place to review:

1. The Software Module Specification as a detailed refinement of Software Design Specification
2. The Software Module Test Specification as a revision of the Software Integration Test Specification.
3. The Software Module Test Specification as a complete and satisfactory statement of testing according to the Software Module Specification.

E.7 Software Design Review Phase

The Software Design Review signals the end of the software and system design and enables the software coding to commence.

The Software Design Review is described in detail in Clause 9.3 of this International Standard.

E.8 Program Development Phase

During the programming of individual modules the Software Module Specification and the Software Module Test Specification are taken as the definitive design documents. In this way the program is prepared against the lowest level of design and with a clearly defined set of tests to which the program is to be tested.

E.9 Software Review and Testing Phase

During this phase of development the source code is reviewed. This review should be supported as far as is possible with analysing tools

Once the code has passed the review then it can be tested against its respective Software Module Test Specification.

Once all the individual programs have satisfied their respective Software Module Test Specifications, they are integrated to form a complete software system.

This complete software system is then tested against the Software Integration Test Specification.

On completion of testing against the Software Integration Test Specification, Software Validation shall take place. Software Validation considers whether the software in total, as represented by source and executable code, software test records, and software documentation, reflects the Software Requirements Specification in such a way that the Software will pass the Software Acceptance Test.

E.10 Software Acceptance Phase

During this phase the Software Acceptance Test is carried out to satisfy the customer that the Software meets the Software Requirements Specification. The phase is omitted if there is no Software Acceptance Test Specification for the project.

E.11 System Testing Phase

During this phase the software and hardware systems in the project are integrated. The complete system according to the System Specification is then subjected to testing according to the System Integration Test Specification.

On completion of testing against the System Integration Test Specification, System Validation shall take place. System Validation considers whether the system in total, as represented by the software, hardware, test records and documentation, reflects the System Requirements Specification in in such a way that the system will pass the System Acceptance Test.

E.12 System Acceptance Phase

During this phase the System Acceptance Test is carried out to satisfy the customer that the completed system meets the System Requirements Specification.

E.13 Site Support

Site Support is an additional phase which covers the installation of the system and any work after the system has been delivered to the customer. In particular any maintenance enhancements to the completed system.

Enhancements have a life cycle similar to the original project and will be documented in a similar way with Requirements Specification Phases etc.

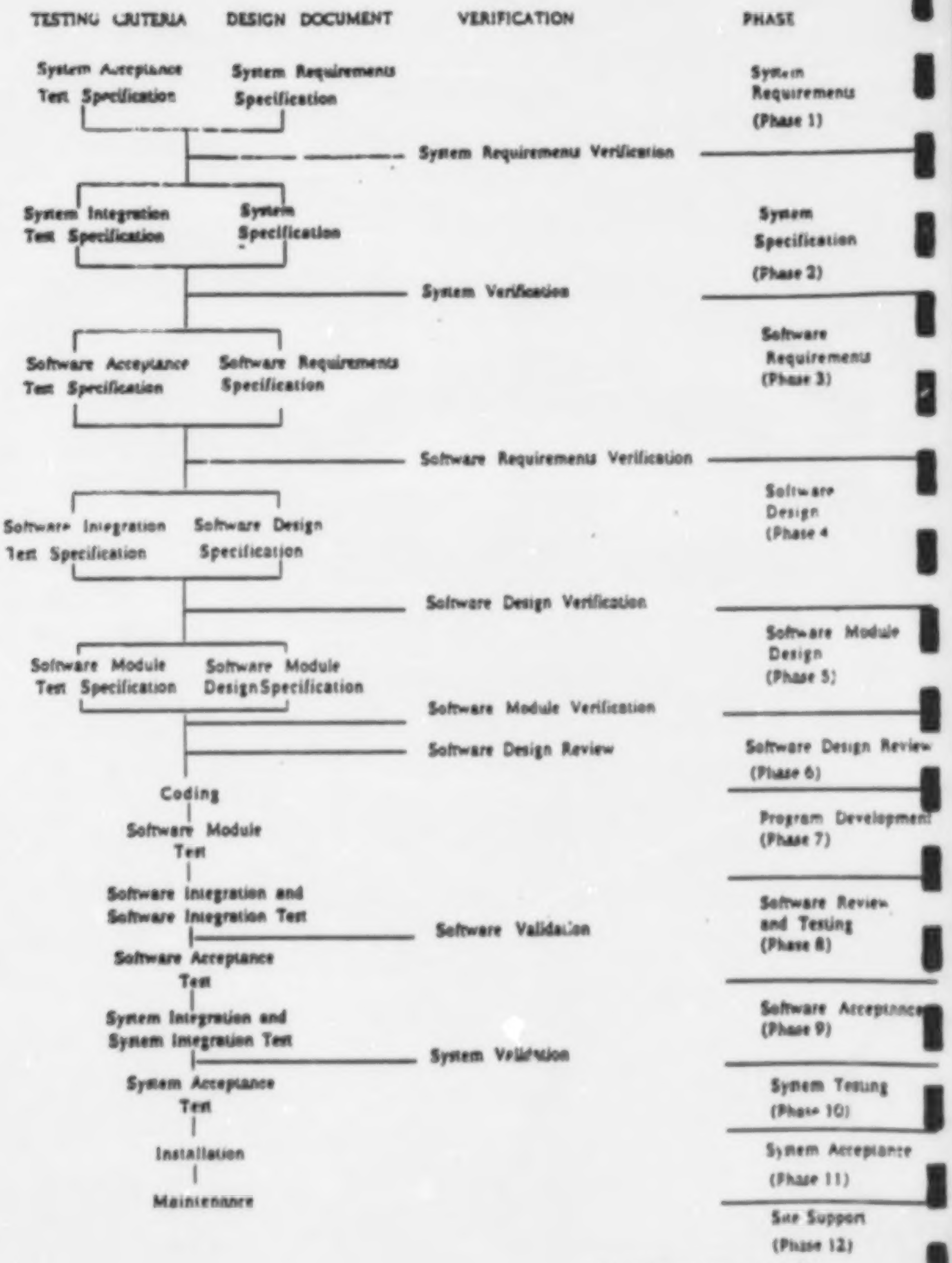
Maintenance documentation is covered in Clause 13 of this International Standard and a specific maintenance cycle is described in Annex F.

E.14 Unscheduled Reviews

In addition to the scheduled verification and validation exercises, unscheduled reviews may be performed at any time. The review must be authorised by the project leader and its objectives must be specified in an Unscheduled Review Specification document.

The procedures used and the personnel involved will be detailed in the Unscheduled Review Plan and should be appropriate to the project phase and review objectives.

Figure E1 - SOFTWARE LIFE CYCLE


TESTING CRITERIA
DESIGN DOCUMENT
VERIFICATION
PHASE
System Acceptance Test Specification
System Requirements Specification
System Requirements (Phase 1)
System Requirements Verification
System Integration Test Specification
System Specification
System Specification (Phase 2)
System Verification
Software Acceptance Test Specification
Software Requirements Specification
Software Requirements (Phase 3)
Software Requirements Verification
Software Integration Test Specification
Software Design Specification
Software Design (Phase 4
Software Design Verification
Software Module Test Specification
Software Module Design Specification
Software Module Design (Phase 5)
Software Module Verification
Software Design Review
Software Design Review (Phase 6)
Coding
Software Module Test
Program Development (Phase 7)
Software Integration and Software Integration Test
Software Validation
Software Review and Testing (Phase 8)
Software Acceptance Test
Software Acceptance (Phase 9)
System Integration and System Integration Test
System Validation
System Acceptance Test
System Testing (Phase 10)
Installation
System Acceptance (Phase 11)
Maintenance
Site Support (Phase 12)

[INFORMATIVE]

ANNEX F

Maintenance

F.1 Introduction

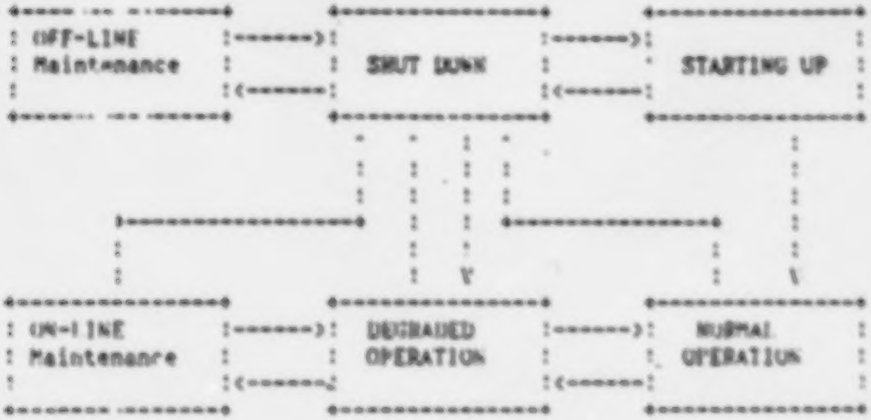
This Annex describes one approach to the maintenance of a software based system in safety related applications which represents good working practice. This Annex has been produced for the instruction and reference of maintenance managers and their staff and for those responsible for the design and implementation of modifications to safety related systems. It is also intended to serve as guidelines to licensing authorities.

F.2 Management, Organisation and Control Procedures

The system (or equipment) as it is considered in this standard can have four typical states and transitions between states (see figure F1).

These states are shut down, starting up, normal operation and degraded operation. Maintenance can be performed on-line and off-line. For each part of the system, subsystem or item of the system there must exist a clear definition of those maintenance operations which can be performed on-line. All other maintenance operations must be carried out off-line.

Fig. F1 : State Transition Diagram for System States and Maintenance



When controlling the maintenance process, the system aspect has to be taken into account (hardware, software, external environment). The safety system is just one of the entities which together with the rest of the system form the environment. When establishing the maintenance and change control policy, initial consideration shall be given to the size and complexity of the environment.

There are four basic components necessary to ensure the success of the maintenance process:

- Safety aspects
- Management techniques
- Technical tools
- Technical expertise

F.2.1. Technical Tools

The technical tools provided to the developer should also be provided to the maintainers. As a guideline the following should be included:

- Spare computing capacity for maintenance tasks, preferably off-line office/developing facilities
- A test environment that is capable of simulating an operational environment.
- Test generators and verifiers (manual or automatic).
- On-line diagnostic programs which offer for example trace, dump inspect and change capabilities.
- Configuration management tools.

F.2.2 Technical Expertise

The typical maintainer should not only be an experienced systems engineer, but also have the analytical skill to determine the cause of a malfunction, the impact of necessary changes and an in-depth knowledge of the application itself, the user requirements and needs and broader aspects of the environment.

F.2.3 Formal Approach to Change Control Procedures

Any change control procedure shall be formal in its design and use. The computer manager shall receive the change request. Since we have to look at the system aspect in safety related applications, the computer manager should receive not only the computer specific change requests, but also change requests from other disciplines. The process of circulating changes to managers of different disciplines lends itself to the need to establish a common modification request document.

Such a document should be simple, effective and designed to enable personnel of any discipline to request a change or modification to the environment. By having the document circulated, other departmental managers can analyse the impact and agree by signing it. Side-effects and the impact of changes on other disciplines can be detected.

Such a procedure needs a multi-level approach:

- A plant or plant item modification request at the top level shall enable change requests to be entered into the environmental management structure to cover the wide range of disciplines involved.
- A software/computer change request at a lower level shall enable details of a computer software change to be managed at the department level.

The lower level document shall be progressed internally at the departmental level. It shall take the original top level request and analyse it.

This analysis is a multi-phase procedure involving:

- Feasibility Study
  - analysis of modification/change request
  - modification resource and cost analysis
  - modification impact analysis
- Detailed specification of the modification/change
- Implementation of the modification/change
- Revalidation and Regression testing of the change and impacted changes.
- Configuration Management.

The multi-phase procedure should be established in the following order:

- receive modification request from user management
- assess impact and feasibility
- analyse and specify modification
- carry out modification
- revalidate
- configuration management
- report to management.

Each step in the multi-phase analysis shall be described in a document. Documentation should be complete, correct and concise. A document shall not be "signed off" without the requisite checking.

The low level document can be initiated without the need for a top-level document. The computer manager shall decide if it is necessary to generate a top-level document.

A maintenance database shall be established. This database can be manual or computer based, preferably the latter. Such a database shall hold:

- all change reqests
- progress of changes
- modification impact list
- test database
- hardware and software item history
- review reports
- system configuration data
- revalidation data
- reliability data

The database shall be incorporated as part of a total configuration management system. Ideally there should be a single database for all identified departments within the environment. Departments should be at least aware of changes made to each other's databases, e.g. changes to interface and hardware circuit diagrams should relate to software changes identified in the software documentation. The database shall be maintained by an appointed individual, for example a Systems Manager who should control access rights.

F.3. Maintenance Activities

F.3.1. The Maintenance Cycle

Maintenance of a computer based system throughout the operational phase of its life cycle can be considered as a sequential series of maintenance cycles.

There are two key activities which shall be performed each time a maintenance cycle is carried out:

- The first is assessing and then carrying out the requested maintenance itself. (Maintenance Action).
- The second is retesting the system and recording the details of the activities carried out and any resulting changes in the system configuration. (Revalidation and Configuration Management).

To achieve good working practices a maintenance database shall be established. This database shall contain all data necessary to carry out the required maintenance together with updated configuration details and historical records.

Maintenance Actions cover all the procedures involved in:

- Preventive maintenance
  Corrective maintenance
  Perfective and/or adaptive maintenance

The maintenance cycle for preventive maintenance shall start when a routine request for preventive maintenance is scheduled from the preventive maintenance database.

The maintenance cycle for corrective maintenance shall start when an anomaly report is raised by the operations staff. The operations staff shall be made responsible for raising the anomaly report irrespective of who it was who first noted the anomalous behaviour of the system.

The maintenance cycle for perfective and/or adaptive maintenance shall start by a modification feasibility request initiated by the user management. If the resulting feasibility report is confirmatory, the user management can then issue a validated modification request and the implementation of the requested modification can proceed.

The initial routine maintenance data in the maintenance database shall be made available by the original supplier of the equipment. During the operational lifetime the system modifications shall be made to the routine maintenance data contained in the maintenance database. These updates to the maintenance data can come from external reports from suppliers, other users or external safety bodies. They may also result from information gathered during earlier maintenance cycles. Modifications shall only be implemented following authorisation by the user management who shall issue a defined modification to the routine maintenance data.

The different forms of maintenance are described on three increasing levels of detail.

F.3.2. Preventive Maintenance

At the highest level, an initial set of routine maintenance data is passed to the maintenance database from the original supplier. Any modifications are channelled or authorised by the user management (see figure F2)

Preventive maintenance cycles are initiated in accordance with the system maintenance schedules contained within the initial routine maintenance data. After completion of the required preventive maintenance a report is issued to the user management detailling the work done and a revalidation request is issued to retest the system. Finally, after carrying out the necessary revalidation, a second and final report is issued to the user management (revalidation and configuration management report). The user management or the operations staff shall decide if the system or subsystem can be returned to service.

During the preventive maintenance cycle, a need may be identified for corrective maintenance. A preventive maintenance failure report shall be raised which shall initiate a corrective maintenance cycle and the procedures appropriate to corrective maintenance shall follow thereafter.

Fig. F2 : Preventive Maintenance. Level 1 and 2.

```
                Initial Routine
SUPPLIER: ---------------------------------------------+
                Maintenance Data                       :
                                                       :
                                                       :
                    Modifications to Routine           :
USER MANAGEMENT: -------------------------------+      :
                         Maintenance Data       :      :
                                                :      :
                                                :      :
                                                V      V
                                             ==============
                                              MAINTENANCE
                                               DATABASE
                                             ==============
                                                   :
                                   Maintenance     :
                                   Action Data     :
                                                   :
                                                   V
            ...............................................
            .  PREVENTIVE MAINTENANCE LEVEL 2      :      .
            .                                      :      .
MAINTENANCE .                                      \      .
ACTION      .   USER         Preventive  +--------------+ .
LEVEL 1     . MANAGEMENT:<---------------: PREVENTIVE   : .
            .               Maintenance  : MAINTENANCE  : .
            .                  Report    +--------------+ .
            .                                 :      :    .
            .                                 :      :    .
            .                   Preventive    :      :    .
            .                  Maintenance    :      :    .
            .                  Failure Report :      :    .
            .                                 :      :    .
            .       +-------------+           :      :    .
            .       : CORRECTIVE  :<----------+      :    .
            .       : MAINTENANCE :                  :    .
            .       +-------------+                  :    .
            .                                        :    .
            .                                        V    .
            ...............................................
                                :                    :
                                :                    :
USER            Maintenance     :       (Routine)    :
MANAGEMENT: <-------------------+      Revalidation  :
               Action Report             Request     :
                                                     V
                                             +---------------+
                                             : REVALIDATION  :
USER                 Revalidation and        :      AND      :
MANAGEMENT: <--------------------------------: CONFIGURATION :
                       Configuration         :   MANAGEMENT  :
                     Management Report       +---------------+
```

On the lowest level three separate activities are identified within the preventive maintenance activity:

- Inspection Testing
- Scheduled Part Replacement
- The carrying out of Servicing

Scheduled Part replacement is carried out routinely during system maintenance schedules or if inspection testing reveals a faulty component.

Inspection Testing is carried out routinely in accordance with the system maintenance schedules. The results of the testing process may be satisfactory or it may identify a need to replace faulty components or to carry out a servicing activity.

Inspection Testing may reveal a need for corrective maintenance. A corrective maintenance cycle shall be initiated by recording this fact in a preventive maintenance failure report.

A Servicing Activity shall be performed regularly in accordance with the system maintenance schedules if identified as necessary following inspection testing

F.3.3 Corrective Maintenance

At the highest level, a corrective maintenance cycle shall start when anomalous behaviour is observed to have occurred within the system and as a result an anomaly report is issued by the operations staff. The necessary maintenance action shall be carried out and a maintenance action report shall be passed to user management (see figure F3).

The necessary maintenance action data should be drawn from the maintenance database. Experience gained in performing corrective maintenance shall result in a recommendation to change the way in which routine maintenance is performed. Any such recommendation shall be passed to the user management as a maintenance data update. It shall be the responsibility of the user management to authorise if they wish the implementation of a modification to the routine maintenance data in the maintenance database.

After completion of the maintenance activities, a revalidation of the system shall be performed and any changes shall be recorded in the maintenance database.

At the next level, the fact that the need for corrective maintenance may arise during the carrying out of preventive maintenance is identified.

Two key activities shall be carried out:

- diagnosis of the problem
- carrying out the necessary corrective modification.

Fig. F3: Corrective Maintenance. Level 1 and 2.

```
                                             ==============
OPERATIONS                                     MAINTENANCE
STAFF:                                          DATABASE
    :                                        ==============
    :                                              :
    :                              Maintenance     :
    :                              Action Data     :
    :                                              V
    :        ..............................................
    :        . CORRECTIVE MAINTENANCE LEVEL 2      :      .
    :        .                                     :      .
    :        .  +---------------+ Preventive       :      .
Anomaly      .  : PREVENTIVE    :---------------+  :      .
Report       .  : MAINTENANCE   : Maintenance   :  :      .
    :        .  +---------------+ Failure Report:  :      .
    :        .                                  V  V      .
    :        .                           +--------------+ .
    +------->---------------------------->:  DIAGNOSIS   : .
             .                   +-------:              : .
MAINTENANCE  .      Maintenance  :       +--------------+ .
ACTION       .      Data Update  :                 :      .
LEVEL 1      .                   :     Corrective  :      .
             .                   :    Modification :      .
             .                   :       Request   :      .
             .                   :                 V      .
             .  USER       Modification  +--------------+ .
             .MANAGEMENT:<---------------: MODIFICATION : .
             .             Result Report :              : .
             .                           +--------------+ .
             .                   :              :    :    .
             .                   V              V    V    .
             ..............................................
                             :   :              :    :
USER            Maintenance  :   :              :    :
MANAGEMENT: <----------------+   :              :    :
               Action Report     :              :    :
                                 :              :    :
                                 :              :    :
USER            Maintenance      :              :    :
MANAGEMENT: <--------------------+              :    :
        :       Data Update                     :    :
        :                                       :    :
        : Modification to                       :    :
        : Routine Maintenance      Configuration:    : Revalidation
        : Database                    Changes   :    :   (Test)
        V                                       :    :   Request
   =============                                :    :
    MAINTENANCE                                 :    :
     DATABASE                                   :    :
   =============                                V    V
                                             +---------------+
                                             : REVALIDATION  :
USER                 Revalidation and        :      AND      :
MANAGEMENT: <--------------------------------: CONFIGURATION :
                       Configuration         :   MANAGEMENT  :
                     Management Report       +---------------+
```

At the third level the diagnosis process consists of two component parts:

- an anomaly diagnosis shall be carried out in order to identify the failed component or subsystem
- with the failed item or items identified, an analysis of the problem shall be carried out to determine the cause of the failure and the form of corrective modification required (failure cause and problem analysis).

If the fault can be rectified without changing the basic configuration of the software or hardware of the system, then an unscheduled part replacement shall be made.

If however corrective modification involves changes to the system configuration, the alternative action of carrying out the configuration modification shall follow. The changes to the configuration shall be documented and recorded. A configuration modification may involve more extensive retesting also.

At regular intervals a component reliability analysis shall be carried out using the information recorded in the failure database. If the results of this analysis reveal a need to alter the routine maintenance schedules, a recommendation to change the routine maintenance schedules should be passed to the user management as a maintenance data update.

1.7.6. Perfective and Adaptive Maintenance

A perfective or adaptive maintenance cycle shall be started by user management by a request to carry out a feasibility study into a proposed modification. It is the responsibility of the user management to issue this request irrespective whoever it was who originated the initial request.

The results of the feasibility study shall be reported back to the user management in a feasibility report. The user management may either reject the proposed modification or decide to proceed further by issuing a validated modification request. A modification shall only be implemented and completed on a validated modification request. All the details shall be reported back to the user management.

After implementation of the modification, a revalidation request shall be issued and the resulting configuration changes shall be passed to the revalidation and configuration management. A final report to the user management shall be carried out after the necessary actions have been taken.

The maintenance action for perfective maintenance and/or adaptive maintenance has two constituent parts, which are

- carrying out a feasibility study
- implementation of a modification.

The implementation of the modification shall only proceed after authorisation by the user management (level 2).

Fig. F6 : Perfective and Adaptive Maintenance, Level 1 and 2.

```
                     Modification Feasibility
USER MANAGEMENT: -------------------------------+
                     Request                    :
                                                :
                                                :
MAINTENANCE ACTION LEVEL 1                      V
            ...............................................
            . PERFECTIVE MAINTENANCE LEVEL 2    :         .
            .                                   :         .
            .                                   V         .
            .                           +---------------+ .
            . Feasibility Report        : FEASIBILITY   : .
  +-------------------------------------:     STUDY     : .
  :         .                           +---------------+ .
  :         .                                      .      .
  :         .                           Maintenance:      .
  :         .                           Action Data:      .
  :         .                                   ===============
  :         .                                     MAINTENANCE
  :         .                                      DATABASE
  :         .                                   ===============
  V         . Validated Modification               :      .
USER   --------------------------------------+     :      .
MANAGEMENT: . Request                        :     : Maintenance
            .                                :     : Action Data
            .                                :     :      .
            .                                V     V      .
            .   USER       Modification  +--------------+ .
  +---------- MANAGEMENT:<---------------: MODIFICATION : .
  :         .              Result Report :              : .
  :         .                            +--------------+ .
Maintenance .                                   :    :    .
Action      .                                   V    V    .
Report      ...............................................
  :                                             :    :
  :                                             :    :
  V                                Configuration:    : Revalidation
USER                                  Changes   :    :   Request
MANAGEMENT                                      :    :
                                                V    V
                                             +----------------+
                                             : REVALIDATION   :
USER                 Revalidation and        :      AND       :
MANAGEMENT: <--------------------------------: CONFIGURATION  :
                       Configuration         :   MANAGEMENT   :
                     Management Report       +----------------+
```

[illegible]
Centre de [illegible]
RECEIVED - [illegible]
JUL 9 [illegible]